

Lossless Compression for μ -Law (A-Law) and IMA ADPCM on the Basis of a Fast RLS Algorithm

Dawei Huang

Abstract— Lossless compression methods are introduced for μ -law (A-law) and IMA ADPCM standards. Lossless compression here means that for a given original input we generate the exactly same output as that of these standards while reducing the bit rates of the compressed files. To keep the same quality, we use the same quantization methods as that in the standards. To reduce the bit rates, we use prediction and entropy coding techniques. The prediction is based on a fast Recursive Least Square (RLS) algorithm which requires less computation than existing RLS algorithms. The entropy coding is based on Huffman coding. The prediction, quantization and coding steps are integrated into an adaptive scheme. Then we can keep the same quality while reducing the bit rates per sample. Comparing to 8 bit per sample μ -law or A-law, only 3.24 bits per sample are needed for coding of audio signal with 44100 Hz sampling frequency and 4.72 bits for speech/audio signals at 11025 Hz. Some improvements for IMA ADPCM standard are also obtained.

I. INTRODUCTION

μ -Law and A-Law in the ITU (CCITT) G.711 were introduced in 1972 for telephony communications. They convert 16 bits per sample digital speech/audio signals into 8 bits per sample by a logarithmic formula. IMA ADPCM was introduced by the Interactive Multimedia Association in 1992. It converts 16 bits per sample digital speech/audio signals into 4 bits per sample. Both standards have been widely used in compression for communications and storage.

The modern compression techniques consist of three approaches: prediction, quantization and entropy coding. In μ -Law and A-Law, a logarithm formula provides the quantization steps. The logarithmic law takes the human perception into account. It also converts a Laplacian distribution, which has a sharp peak around zero, of the original signals into a much more flat distribution. So the unified coding scheme (8 bits for all 256 states) already closes to the optimal entropy coding. In contrast to ITU G.721/723/G.726 ADPCM standards, IMA ADPCM uses a simple difference for prediction. Then an adaptive quantization scheme is introduced to convert 16 bits (65536 states) original signals into 4 bits (16 states) codes. However, if we plot the histogram of the IMA ADPCM codes, we will find that the histogram is not flat, i.e., the frequencies of those 16 states are much different. So, from the entropy coding point of view, IMA ADPCM is not optimal. To confirm this, one can use an entropy coding algorithm, like Lample-Ziv algorithm in WinZip, to re-compress IMA ADPCM files. Usually the Lample-Ziv algorithm can reduce the IMA ADPCM bit rate from 4 to around 3.6 bits

per sample.

In this paper, we introduce an integrated scheme of prediction, quantization and entropy coding. This scheme uses the same quantization methods as that in the standards while reducing the bit rates by more sophisticated prediction and entropy coding methods. In contrast to the lossless compression method for PCM in [8], the scheme is adaptive. In [8] AR(2) models or differences were used in each 256 sample block for prediction. The coefficients of the AR models are calculated by the data in the block and are fixed within each block. Then both the coefficients and the residuals are coded for storage or transmission. In our scheme, the residual is calculated by the previous predictor. Then the predictor is updated based on the observation up to that time. So we do not need coding the prediction coefficients. As long as the past signal in decoding side is the same as that in the encoding side, we can synchronize the next signal and then the next predictor in both encoding and decoding sides. The benefit of this scheme is that we can use a large number of the coefficients in the predictor without increasing the code length. Since voice signal has a long dependence, especially for the CD quality sound with 44,100 Hz sample frequency, the long regression increases the efficiency of the prediction and then reduces the bit rates for coding the residuals.

We use a fast Recursive Least Squares (RLS) algorithm for prediction. RLS algorithm has been used for a long time. It was designed in a Kalman filter form which requires $O(q^2)$ multiplications and divisions [1]. Here and after q is the number of tap coefficients in the predictor. Recently, several fast RLS algorithms [2-3] have been introduced to reduce the number of multiplications and divisions to $O(q)$. The result in [3] requires $17q$ multiplications and divisions. Another drawback of existing RLS algorithms is its stability. Since we have to deal with ill-conditioned matrices, RLS may not be able to run for a large number of samples. So RLS/LMS switching scheme is usually adopted for speech/audio signal prediction [4]. However, after a minor modification, Theorem 5-1 in [5] can be used for deriving RLS recursion. It only needs $7q$ multiplications and divisions. Also, this algorithm can be applied for up to 5000 sample blocks without the problem in stability. More flexible recursive algorithms, including the cases that q increases or decreases, were shown in [6, 7]. We show the basic idea and give the whole formulae in this paper for reader's convenience.

In Section 2 the RLS algorithm is introduced and compared with the existing methods. An integrated adaptive scheme of prediction, quantization and coding for lossless

Queensland University of Technology, email: daweih@hotmail.com or d.huang@fsc.qut.edu.au

compression of μ -Law (A-Law) and IMA ADPCM is introduced in Section 3. A Windows platform program has been produced on the basis of these algorithms. It can be performed to encode and decode the speech/audio signals on-line and in real time for up to 44,100 samples per second (CD standard) on a notebook computer with Pentium 233 kHz CPU. The software and its demonstrations are introduced in Section 4.

II. A FAST RECURSIVE LEAST SQUARES (RLS) ALGORITHM

For a given forgetting factor $0 < \lambda \leq 1$, the RLS algorithm updates the prediction coefficients $A_t = [a_1(t), \dots, a_q(t)]$ which minimizes the following target function:

$$\sum_{k=q}^t \lambda^{t-k} \left(x_k - \sum_{j=1}^q a_j(t) x_{k-j} \right)^2,$$

where $\{x_t, t = 0, 1, \dots\}$ is the input signal time series.

The direct solution of this problem can be obtained by solving a linear equation system. However, it requires $O(q^3)$ multiplications in each update time. Using Kalman filter, one can derive the following algorithms (see, for example, [1]):

$$\begin{aligned} A_t &= A_{t-1} + (x_t - \phi_t' A_{t-1}) V_t \phi_t, \\ V_t &= \left(V_{t-1} - \frac{1}{\lambda + \phi_t' V_{t-1} \phi_t} V_{t-1} \phi_t \phi_t' V_{t-1} \right) / \lambda, \end{aligned} \quad (1)$$

where $\phi_t = [x_{t-1}, x_{t-2}, \dots, x_{t-q}]'$. It requires $O(q^2)$ multiplications. Recently, a variety of algorithms has been developed to reduce the calculation to $O(q)$. For example, QR decomposition of matrices is used in [2]. Also, UD factorization of matrices is applied in [3] to reduce the calculation down to $17q$ multiplications in each run.

In this section, we derive an algorithm with only $7q$ multiplications in each update time. Our approach is based on the projection in an Euclidean space. We explain the Least Squares problem in projection and show some basic properties of projections first. Then we introduce two supplementary projections and establish equations for them. Finally, we derive the whole algorithm.

A. Projection in Euclidean space

To construct a recursive structure, we modify this target function to

$$\sigma_q^2(t) \equiv \min_{a_j(t)} \sum_{k=0}^t \lambda^{t-k} \left(x_k - \sum_{j=1}^q a_j(t) x_{k-j} \right)^2, \quad (2)$$

where $x_k = 0$ for all $k < 0$. The least squares problem can be explained as projection in an Euclidean space as the following: Let

$$\vec{x}_{t,m} = \left[x_{t-m}, \lambda x_{t-m-1}, \dots, \lambda^{t-m} x_0, \underbrace{0, \dots, 0}_m \right]',$$

$$\begin{aligned} & X_m^n(t) \\ &= \left[\vec{x}_{t,m}, \vec{x}_{t,m+1}, \dots, \vec{x}_{t,n} \right]' \\ &= \begin{bmatrix} x_{t-m} & x_{t-m-1} & \dots & x_{t-n} \\ \lambda x_{t-m-1} & \lambda x_{t-m-2} & \dots & \lambda x_{t-1-n} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda^{t-n} x_n & \lambda^{t-n} x_{n-1} & \dots & \lambda^{t-n} x_0 \\ \lambda^{t-n+1} x_{n-1} & \lambda^{t-n+1} x_{n-2} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ \lambda^{t-1} x_1 & \lambda^{t-1} x_0 & \dots & 0 \\ \lambda^t x_0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (3) \end{aligned}$$

$$A_t = [a_1(t), \dots, a_q(t)]',$$

then the least squares problem (2) is equivalent to finding the projector

$$P_{X_1^q(t)} \vec{x}_{t,0} = X_1^q(t) A_t, \quad \sigma_q^2(t) = \left\| \vec{x}_{t,0} - P_{X_1^q(t)} \vec{x}_{t,0} \right\|^2, \quad (4)$$

here and after $P_{X_m^n(t)} \vec{y}$ denotes the projector of a t -dimensional vector \vec{y} onto the Euclidean space spanned by the column vectors in the matrix $X_m^n(t)$, $\|\vec{y}\|^2 = \vec{y}'\vec{y}$.

Without loss of generality, we can assume that $x_0 \neq 0$. Then one can confirm that the column vectors in $X_m^n(t)$ are linearly independent, i.e., the matrix $[X_m^n(t)]' X_m^n(t)$ must be positive definite, though its minimum eigenvalue may tend to zero as t tends to infinity when $\lambda < 1$.

The basic tool for calculating projections we use is the following lemma [5, 7]:

Lemma 1: Let X, Z be vectors and Y be a matrix, then

$$P_{[Y,Z]} = P_Y + P_{[Z-P_Y Z]}, \quad (5)$$

$$\begin{aligned} & \left\| X - P_{[Y,Z]} X \right\|^2 \\ &= \left\| X - P_Y X \right\|^2 - \left\| P_{[Z-P_Y Z]} X \right\|^2 \end{aligned} \quad (6)$$

Further,

$$P_{X_m^n(t)} + P_{[U_t - P_{X_m^n(t)} U_t]} = P_{\lambda X_m^n(t-1)} + P_{U_t}, \quad (7)$$

$$\begin{aligned} \left\| \vec{x}_{t,k} - P_{X_m^n(t)} \vec{x}_{t,k} \right\|^2 &= \lambda^2 \left\| \vec{x}_{t-1,k} - P_{X_m^n(t-1)} \vec{x}_{t-1,k} \right\|^2 \\ &+ \left\| P_{[U_t - P_{X_m^n(t)} U_t]} \vec{x}_{t,k} \right\|^2. \end{aligned} \quad (8)$$

B. Two supplementary projectors

In (1), calculating $V_t \phi_t$ needs q^2 multiplications. Thus, to get computation down to $O(q)$, we have to use a supplementary coefficient vector

$$C_{t-1} = [c_1(t-1), \dots, c_q(t-1)]' \triangleq V_t \phi_t$$

Then, in contrast to updating V_{t-1} in the Kalman filter, we need to find formula to update C_{t-1} .

Firstly, similar to A_t , we explain $\{c_1(t-1), \dots, c_q(t-1)\}$ as projection coefficients. In fact, let

$$U_t = [1, \underbrace{0, \dots, 0}_{t-1}]^T,$$

then

$$P_{X_1^q(t)} U_t = X_1^q(t) C_{t-1}.$$

Now, using the special structure in (3), one can conclude that

$$P_{X_0^{q-1}(t-1)} U_{t-1} = P_{X_1^q(t)} U_t.$$

Thus, updating C_{t-1} to C_t is equivalent to finding $P_{X_0^{q-1}(t)} U_t$ on the basis of $P_{X_1^q(t)} U_t$ and $P_{X_1^q(t)} \vec{x}_{t,0}$.

It follows from (5) that

$$\begin{aligned} & P_{X_1^q(t)} U_t + P_{[\vec{x}_{t,0} - P_{X_1^q(t)} \vec{x}_{t,0}]} U_t \\ &= P_{X_0^q(t)} U_t \\ &= P_{X_0^{q-1}(t)} U_t + P_{[\vec{x}_{t,q} - P_{X_0^{q-1}(t)} \vec{x}_{t,q}]} U_t. \end{aligned} \quad (9)$$

Thus, we need another supplementary projector

$$P_{X_0^{q-1}(t)} \vec{x}_{t,q} = X_0^{q-1}(t) B_t,$$

where $B_t = [b_1(t), \dots, b_q(t)]'$.

Since vectors $\{\vec{x}_{t,0}, \vec{x}_{t,1}, \dots, \vec{x}_{t,q}\}$ are linear independent, (9) gives q linear equations on the $2q$ unknown coefficients in C_t and B_t .

A quantity in (9) is

$$\begin{aligned} h_t &= [\vec{x}_{t,0} - P_{X_1^q(t)} \vec{x}_{t,0}]' U_t \\ &= x_t - [x_{t-1}, \dots, x_{t-q}] A_t. \end{aligned}$$

However, we can reduce the q multiplications to one if we know

$$\begin{aligned} v_q^2(t-1) &= \left\| U_{t-1} - P_{X_0^{q-1}(t-1)} U_{t-1} \right\|^2 \\ &= 1 - [x_{t-1}, \dots, x_{t-q}] C_{t-1}, \end{aligned}$$

since

$$h_t = d_t v_q^2(t-1).$$

Then we need a formula to update $v_q^2(t)$ from $v_q^2(t-1)$ and $\sigma_q^2(t)$. This formula can be derived by solving an equation established by (6) and (8).

On the other hand, it follows from (7) that

$$\begin{aligned} & P_{X_0^{q-1}(t)} \vec{x}_{t,q} + P_{\{U_t - P_{X_0^{q-1}(t)} U_t\}} \vec{x}_{t,q} \\ &= P_{\lambda X_0^{q-1}(t-1)} \lambda \vec{x}_{t-1,q} + P_{U_t} \vec{x}_{t,q}, \end{aligned} \quad (10)$$

Combining (9) and (10), we can calculate B_t and C_t .

C. Recursive algorithm

Let

$$d_t = x_t - [x_{t-1}, \dots, x_{t-q}] A_{t-1},$$

then

$$\begin{aligned} A_t &= A_{t-1} + d_t C_{t-1}, \\ \sigma_q^2(t) &= \lambda^2 \sigma_q^2(t-1) + d_t^2 v_q^2(t-1). \end{aligned}$$

Also, let

$$\begin{aligned} h_t &= d_t v_q^2(t-1), \\ e_t &= x_{t-q} - [x_t, \dots, x_{t-q+1}] B_{t-1}, \\ f_t &= c_q(t-1) - a_q(t) d_t^2 v_q^2(t-1) / \sigma_q^2(t), \\ K_t &= 1 - e_t f_t, \end{aligned}$$

then

$$\begin{aligned} c_1(t) &= (f_t b_1(t-1) + h_t) / K_t, \\ c_j(t) &= [f_t b_j(t) + c_{j-1}(t-1) - h_t a_{j-1}(t)] / K_t, j = 2, \dots, q, \\ v_q^2(t) &= \lambda^2 v_q^2(t-1) \sigma_q^2(t-1) / [\sigma_q^2(t) K_t]. \\ b_j(t) &= e_t c_j(t), j = 1, \dots, q. \end{aligned}$$

III. ADAPTIVE LOSSLESS COMPRESSION

In contrast to the lossless compression for PCM in [8], to synchronize the encoding and decoding sides in a compression scheme with quantization, we must predict the recovered signal instead of the original signal. In fact, all quantization based coding scheme can be described by the following formula:

$$q_t = Q(x_t), \quad r_t = L(q_t).$$

Here and after $\{x_t\}$ is the input signal, $Q(\bullet)$ is a multi-to-one mapping quantization function, $\{q_t\}$ is the corresponding code sequence, $\{r_t\}$ is the recovered signal and $L(\bullet)$ is an one-to-one mapping. Since $x_t \neq r_t$ in general and only $\{r_t\}$ is available on the decoding side, we apply the RLS algorithm introduced in Section 2 to $\{r_t\}$.

Starting from a set of initial values $\{r_{-q}, \dots, r_0\}$, which is the same on both encoding and decoding sides, we do the following recursion:

A. Lossless encoding for μ -law and A-law

Step 1: Obtain the output

$$q_t = Q(x_t), \quad r_t = L(q_t).$$

Step 2: Prediction in the signal domain

$$\hat{r}_t = [r_{t-1}, r_{t-2}, \dots, r_{t-q}] A_{t-1}$$

and the code domain

$$\hat{q}_t = Q(\hat{r}_t).$$

Step 3: Update the prediction coefficients

$$\begin{aligned} A_t &= A_{t-1} + (r_t - \hat{r}_t) C_{t-1}, \\ C_t &= f_1(A_t, B_{t-1}, C_{t-1}, r_t, \dots, r_{t-q}), \\ B_t &= f_2(A_t, B_{t-1}, C_{t-1}, r_t, \dots, r_{t-q}). \end{aligned}$$

Here the functions f_1 and f_2 are given in Section 2.

Step 4: Find the difference of transforms of q_t and \hat{q}_t .
For a μ -law or A-law code y in 8 bits, let

$$\begin{aligned} T_1(y) &= \begin{cases} y, & y < 128 \\ 383 - y, & y \geq 128. \end{cases} \\ T_2(y) &= \begin{cases} 256 + y, & y < -128, \\ y, & -128 \leq y < 128, \\ y - 256, & y \geq 128. \end{cases} \quad (11) \end{aligned}$$

Then

$$d_t = T_2(T_1(q_t) - T_1(\hat{q}_t)).$$

Step 5: Huffman encoding

$$c_t = H(d_t).$$

B. Lossless compression for IMA ADPCM

IMA ADPCM adopts an adaptive step s_t for quantization. Using the same notation above, we can describe it by the following:

$$\begin{aligned} q_t &= Q(x_t - r_{t-1} | s_{t-1}), \quad s_t = S(q_t | s_{t-1}), \\ r_t &= L(q_t | s_{t-1}). \end{aligned} \quad (12)$$

Here q_t is a integer between 0 and 15.

The same 5 step recursion for μ -law and A-law can be used provided that we replace $Q(\bullet)$, $L(\bullet)$, $T_1(\bullet)$ and $T_2(\bullet)$ in the 5 steps by (12) and the following formula:

$$\begin{aligned} T_1(y) &= \begin{cases} y, & y < 8 \\ 7 - y, & y \geq 8. \end{cases} \\ T_2(y) &= \begin{cases} 16 + y, & y < -8, \\ y, & -8 \leq y < 8, \\ y - 16, & y \geq 8. \end{cases} \end{aligned}$$

IV. SOFTWARE AND DEMONSTRATIONS

A Windows based platform with μ -law, A-law and IMA ADPCM compression programs as the engine, all were written in Visual C++, was produced. This software can download, play, encode, decode and reproduce sound with wav format. Large number of samples were tested. Some results were presented in the following table. The audio sample were taken from Compact Disks for pieces of symphony, whole songs, etc. The speech sample was 5 minute news of Voice of America downloaded from the Internet. The speech/audio samples are broadcasting of Voice of America with possible music background.

Samples Audio	Hz	Lasting Time in seconds	Lossless μ -law	Lossless ADPCM
1	44,100	246	2.70	2.59
2	44,100	88	3.09	2.69
3	44,100	200	3.27	2.73
4	44,100	163	3.33	3.04
5	44,100	289	3.35	2.74
6	44,100	146	3.48	2.69
7	44,100	220	3.53	2.79
Average Speech/Audio			3.24	2.75
1	11,025	301	4.36	3.08
2	11,025	601	4.63	3.19
3	11,025	600	4.70	3.22
4	11,025	600	4.74	3.15
5	11,025	1765	4.82	3.17
Average			4.72	3.17

From the table one can see that the software works better for audio. This is because the audio samples are taken in 44,100 Hz. So their prediction is only for 1/44,100 second ahead and is easier than that of speech samples in 11,025 Hz. Also, purely speech looks like easier to be compressed than that in a music background.

References:

- [1]: S. Haykin (1996), Adaptive filter theory, Prentice Hall.
- [2]: I.K. prouder (1994), Fast time-series adaptive-filtering algorithm based on the QRD inverse-updates method, Proc. Inst. Elec. Eng. Vis. Image signal Process, vol. 141, pp325-332.
- [3]: A. Carini and E. Mumolo (1999), A numerically stable fast RLS algorithm for adaptive filtering and prediction based on the UD factorization, IEEE Trans. Signal Processing, Vol. 47, No. 8, pp2309-2313.
- [4]: E. Mumolo, M. Scagnol and A. Carini (1998), An algorithm for wideband audio coding based on LMS/RLS switched predictors, In Proc. IX Euro. Signal Process. Conf., Rodos, Greece, Sept. 8-11.
- [5]: D. Huang (1988), Recursive method for ARMA model estimation (I), Acta Mathematicae Applicatae Sinica, Vol. 4, No. 2, pp169-192.
- [6]: D. Huang (1989), Recursive method for ARMA model estimation (II), Acta Mathematicae Applicatae Sinica, Vol. 5, No. 4, pp332-354.
- [7]: D. Huang (1989), Levinson-type recursive algorithms for least-squares atoregression, Journal of Time Series Analysis, Vol. 11, No. 4, p295-315.
- [8]: T. Robinson (1996) SHORTEN: Simple lossless and near-lossless waveform compression, Proceedings ICASSP 1996.