LOSSY COMPRESSION OF INDIVIDUAL SIGNALS BASED ON STRING MATCHING AND ONE PASS CODEBOOK DESIGN

Christopher Chan and Martin Vetterli

University of California
231 Cory Hall
Berkeley, CA 94720
wychan,martin@eecs.berkeley.edu

ABSTRACT

This paper describes an effort to extend the Lempel-Ziv algorithm to a practical universal lossy compression algorithm. It is based on the idea of approximate string matching with a rate-distortion $(R-D)$ criterion, and is addressed within the framework of vector quantization (VQ) [4]. A practical one pass algorithm for VQ codebook construction and adaptation for individual signals is developed which assumes no prior knowledge of the source statistics and involves no iteration. We call this technique rate-distortion Lempel-Ziv (RDLZ). As in the case of the Lempel-Ziv algorithm, the encoded bit stream consists of codebook (dictionary) updates as well as indices (pointers) to the codebook. The idea of "trading" bits for distortion in modifying the codebook will be introduced. Experimental results show that, for Gaussian sources as well as real images, RDLZ performs comparably, sometimes favorably, to static codebook VQ trained on the corresponding sources or images.

1. INTRODUCTION

Dictionary coding achieves compression by replacing blocks of source symbols by indices to some dictionary. The dictionary is composed of blocks of source symbols, or *phrases*, that are expected to occur frequently [1]. In adaptive lossless dictionary coding such as the Lempel-Ziv algorithm, the dictionary is built from phrases already seen in the source sequence, and subsequent occurrences of identical phrases are encoded by their indices to the dictionary.

To extend this idea to lossy compression, phrases from the source are matched to the dictionary with a fidelity criterion. Along these lines, Steinberg and Gutman proposed an algorithm that achieves a rate of $R(D/2)$ given an average distortion $D > 0$ for a large class of sources and distortion measures [8]. Koga and Arimoto further proved that the algorithm achieves the rate-distortion bound asymptotically for certain sources and fidelity criteria [6]. Hence, there is theoretical foundation and motivation for a Lempel-Ziv type of universal lossy compression algorithm.

In our work, this problem is addressed in a VQ context. Building the dictionary has the same flavor as constructing a VQ codebook on-line. Approximate string matching is similar to quantizing to the "best" code-vector, with respect to the Lagrangian $R + \lambda D$. More precisely, code-vectors are learned from the source to be encoded, and will be modified to adapt to variations of the source distribution. No training sets or iteration are involved. In the encoded bit stream, side information about the evolution of the codebook is inserted between sequences of codebook indices. Therefore the decoder can reconstruct an identical copy of the codebook. The construction and adaptation of the codebook, entropy coding of the code-vector indices, and complexity issues will be discussed in Section 2. Results on the performance of RDLZ as compared to static codebook VQ will be presented in Section 3.

Similar works on adaptive quantization include [5] which modifies code-vectors based on their partial distortions, [9] based on the "gold-washing" method, [3] which refines the codebook at increasing intervals, and [7] which assumes no side information and only considers scalar quantization.

2. PRACTICAL ONE PASS ALGORITHM

2.1. Constructing the Codebook

Let $\mathbf{x} = x_1^\ell, x_2^\ell, \ldots, x_N^\ell$, $x_i^\ell \in A^\ell$, be a sequence of source vectors of dimension ℓ. The source alphabet A may be continuous. Let $d_\ell(x^\ell, y^\ell)$ be a distortion measure on $A^\ell \times \hat{A}^\ell$, where $\hat{A}$ is the reconstruction alphabet. The codebook $\mathbf{C}$ is a set of reconstruction code-vectors $\{c_j^\ell \in \hat{A}^\ell, j = 1, \ldots, M\}$, each of which is associated with a cell $C_j = \{x_i^\ell | \hat{x}_i^\ell = c_j^\ell\}$, where $\hat{x}_i^\ell$ is the quantized version of x_i^ℓ, and an index of rate $r(c_i^\ell)$ (length of the index codes in bits).

0-7803-2431-5/95 $4.00

In [2], Chou *et al.* introduced an iterative algorithm called entropy-constrained vector quantization (ECVQ) to design vector quantizers that minimize average distortion subject to an entropy constraint: i) Given $\mathbf{C}$, each x_i^ℓ is quantized to $\hat{x}_i^\ell = c_k^\ell$ such that $r(c_k^\ell) + \lambda d_\ell(x_i^\ell, c_k^\ell)$ is minimized. ii) Then, each c_j^ℓ is updated to the centroid of C_j, and $r(c_j^\ell) = log_2(1/\mathcal{P}(C_j))$ is the new rate of the code-vector index. The algorithm iterates these two steps until convergence is reached.

For a stationary ergodic source, $\mathbf{x}$ can be divided into blocks of length L, i.e. $(x_1^\ell \ldots x_L^\ell), (x_{L+1}^\ell \ldots x_{2L}^\ell)$, etc. Steps i) and ii) can be applied to these blocks in turn, and convergence to the optimal quantizer as L and the number of L-blocks get large is expected.

However, if the stationarity assumption of the source does not hold, at least within a finite time frame, such a one pass block updating scheme may not give the asymptotically optimal quantizer. We have hence studied the effect of adapting the codebook by "trading" bit rates for distortion, through adding, splitting, deleting and updating (towards centroids) code-vectors.

Initial Codebook

Assume the codebook $\mathbf{C}$ is initially empty, i.e. $M = 0$. Let $\tau_j, j = 1, 2, \ldots$ be the time intervals during which the j^{th} L-block of the source is being parsed in. During τ_1, each source vector x_i^ℓ is either encoded by an existing code-vector $c_j^\ell, j = 1, \ldots, M$, or added to the codebook as c_{M+1}^ℓ (quantized onto $\hat{A}^\ell$ if $A \neq \hat{A}$). This can be done by comparing with a distortion threshold D_0 (given as a parameter):

> IF $d_\ell(x_i^\ell, c_k^\ell) < D_0$ for some $k \in \{1, \ldots, M\}$,
> THEN x_i^ℓ is mapped to the nearest c_k^ℓ, ELSE
> add x_i^ℓ to codebook.

An initial codebook is constructed. It is clearly suboptimal, but the rest of the algorithm will correct this. In practice, the initial codebook can be given *a priori* to avoid the overhead in encoding it, which can be substantial.

Updating Code-vectors

At the end of each τ_j, $\{c_i^l\}$ can be updated towards the centroids of C_i calculated during τ_j, as is done in the iterative algorithm described above. However, since bits are spent to encode the code-vector updates, the potential reduction in distortion must be large enough to make this worthwhile.

Let B_u bits be required to encode the difference between the centroid and the original code-vector. Since not all code-vectors may be updated, the code-vector index must be specified also. The average overhead to send a code-vector update (per source vector mapped to it) is

$$\Delta R \doteq \frac{r(c_i^\ell) + B_u}{n(c_i^\ell)},$$

where $n(c_i^\ell)$ is the count of occurrences of c_i^ℓ during τ_i.

For $d_\ell(x^\ell, y^\ell) = \|x^\ell - y^\ell\|_2$, the reduction in average distortion for that code-vector cell is

$$\Delta D = d_\ell(c_i^\ell, \tilde{c}_i^\ell),$$

where $\tilde{c}_i^\ell$ is the centroid.

A code-vector is updated if

$$\lambda \Delta D > \Delta R. \tag{1}$$

Adding Code-vectors

During $\tau_j, j > 1$, there may be times when a source vector x_i^ℓ is too far from any existing code-vector. This may reflect a change in the source distribution, or simply a rare event. In any case, a prediction $\hat{n}$ of the "popularity" of the new code-vector is involved.

A new code-vector requires a code for its index. As will be described in Section 2.2, the indices are entropy coded at the end of each τ_j, based on $n(c_i^\ell), i = 1, \ldots, M$, gathered during τ_j. Assuming that they are prefix-free codes, one code can split into two by expanding it by one bit (visualize splitting a leaf node of the Huffman tree).

Let B_a bits be required to encode the difference between the new code-vector x_i^ℓ with the closest existing code-vector c_k^ℓ. Hence,

$$\Delta R \doteq \frac{n(c_k^\ell) + r(c_k^\ell) + B_a}{\hat{n}},$$

and

$$\Delta D \doteq d_\ell(c_k^\ell, x_i^\ell),$$

where x_i^ℓ may need to be quantized onto $\hat{A}^\ell$.

With these ΔR and ΔD, the add criterion is (1).

Splitting Code-vectors

In [5], it is proposed that the code-vector with the highest *partial distortion* should split. Let $D(C_i, c_i^\ell)$ be the average distortion of C_i with respect to c_i^ℓ. The partial distortion is then defined to be $\mathcal{P}(C_i)D(C_i, c_i^\ell)$. In our work, since the criterion is to minimize $R + \lambda D$, a different approach is taken.

Adding a new code-vector also leads to lengthening of an existing index code by 1 bit. Hence,

$$\Delta R \doteq 1 + \frac{r(c_i^\ell) + B_a}{n(c_i^\ell)}.$$

It is difficult to estimate ΔD, since it depends on the geometry of the code-vectors. A heuristic estimation is given by

$$\Delta D \doteq K D(C_i, c_i^\ell),$$

where K is a constant describing the fractional reduction in distortion by splitting the code-vector.

Unsurprisingly, the splitting criterion is again (1).

Deleting Code-vectors

In [2], code-vectors whose cells are unpopulated after an iteration are effectively deleted from the codebook. For our purpose, it may be worthwhile to keep the code-vectors around, since adding them later on will involve considerable overhead. At the end of each τ_j, code-vectors with $n(c_i^\ell) = 0$ will be assigned a count of 1 before updating the index codes for τ_{j+1}. An exception for this is when the codebook is already too large, in which case some of the unpopulated code-vectors are deleted.

2.2. Entropy Coding of the Codeword Indices

As mentioned before, codes for code-vector indices are updated at the end of τ_i based on $n(c_i^\ell)$ gathered during τ_i. They will be in effect during τ_{i+1}. Since both the encoder and decoder have access to $n(c_i^\ell)$, updates of the index codes do not have to be transmitted. Entropy codes such as Huffman code, Shannon-Fano-Elias code, or arithmetic code can be used for this purpose.

Encoding of the source is done in a manner similar to Lempel-Ziv coding: each source vector is encoded by either a new code-vector or an index to the codebook. A new code-vector is described by the index of the nearest existing code-vector and the differential, which can be coded by a fixed-rate scalar quantizer. At the end of each τ_i, code-vector updates are transmitted likewise. Of course, escape sequences to distinguish side information from code-vector indices are necessary, and their overhead should be considered within the design loop. Deletion can be done synchronously at both ends without side information.

2.3. Complexity of the Algorithm

Finally, a brief note on the algorithmic complexity. Encoding each source vector involves computing M $d_\ell(\cdot,\cdot)$ operations (one for each code-vector) and finding the minimum. Evaluating whether adding a new code-vector is worthwhile involves one comparison between the lowest $d_\ell(\cdot,\cdot)$ and a threshold (computed at the end of each τ_i). Also, several book-keeping operations, such as incrementing $n(c_i^\ell)$, are involved during each encoding step.

Once every L vectors, new entropy codes for the code-vector indices are computed. Code-vector deletion, updating towards centroid, and threshold computations are also performed. As L becomes large, these computational overheads become insignificant. Note that L should be large enough so that accurate statistics can be estimated, and small enough so that the codebook can adapt to short-term non-stationarity.

Hence, RDLZ is of the same order in complexity ($O(M\ell)$ per source vector) as static codebook VQ in terms of source encoding and decoding, except for tree-structured vector quantization (TSVQ) [4], which has a lower complexity. However, the average codebook size M for RDLZ can be kept much smaller.

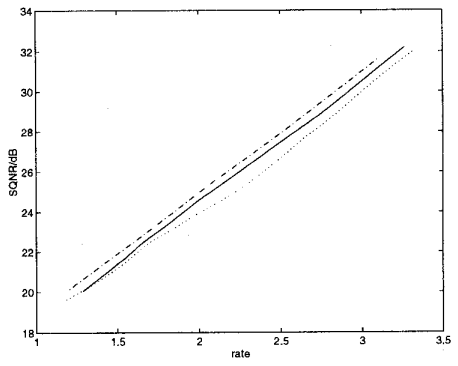

Figure 1: Performance of ECVQ, TSVQ and RDLZ on stationary memoryless Gaussian signals. - - - ECVQ, ··· TSVQ, — RDLZ

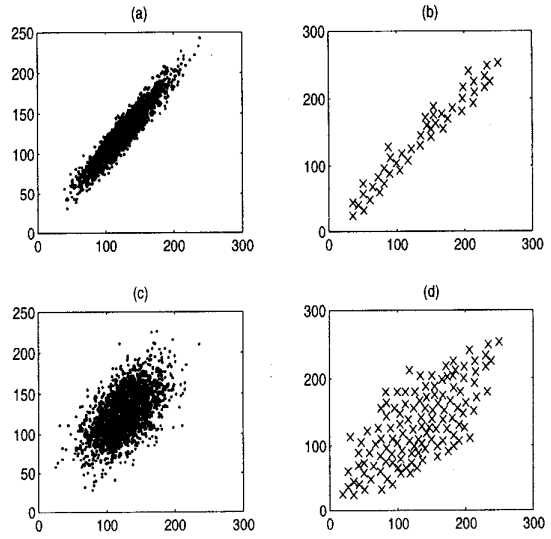

Figure 2: Adaptation of RDLZ Codebook. (a) Source distribution at time t_1, $\rho = 0.9$, (b) codebook at time t_1, (c) source distribution at time t_2, $\rho = 0.6$, (d) Codebook at time t_2. Results: RDLZ gives SQNR = 27.1 dB, rate = 2.3986; TSVQ gives SQNR = 27.1 dB, rate = 2.683

Figure 3: Lenna encoded by RDLZ with initial codebook designed on Barbara. PSNR = 30.84 dB, rate = 0.5142 bpp. TSVQ gives PSNR = 30.77dB, rate = 0.6854 bpp.

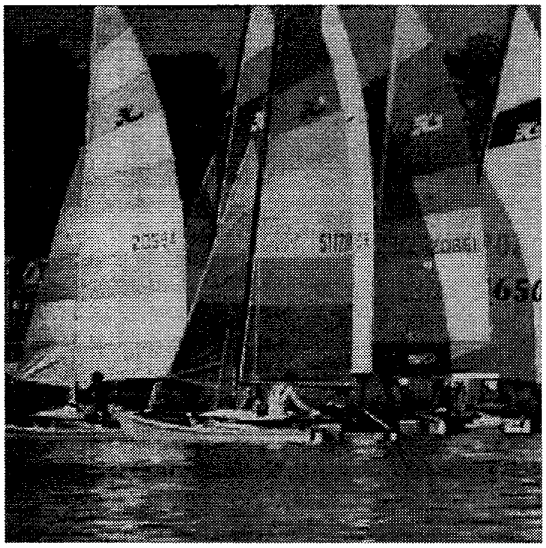

Figure 4: Yacht encoded by RDLZ with the same initial codebook. PSNR = 30.32 dB, rate = 0.5468 bpp. TSVQ gives PSNR = 30.96 dB, rate = 0.5550 bpp.

3. EXPERIMENTAL RESULTS

We have performed experiments on signals generated by stationary ergodic sources. Figure 1 shows the performance of RDLZ on a memoryless Gaussian signal of 120,000 samples as compared to TSVQ and ECVQ. From the results, it can be concluded that RDLZ performs nearly as good as ECVQ and better than TSVQ except for very low bit rates.

For non-stationary sources, RDLZ will outperform static codebook VQ. Figure 2 shows the RDLZ codebook at two different instants. Here, the source is a correlated Gaussian source with varying correlations (ρ from 0.9 to 0.6). Vectors of dimension 2 are used, so that they can be conveniently plotted on the Cartesian plane. The codebook is clearly adapting to changes in the source statistics.

The algorithm was also tested on real images using vectors of dimension 4×4. An initial codebook of size 1000 was designed on the image Barbara as described in Section 2.1. It is then used to code Lenna (Figure 3) and another image of a different nature (Figure 4). Results of TSVQ (trained for the corresponding images) are included for comparison.

4. REFERENCES

[1] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[2] P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-constrained vector quantization. *IEEE Trans. Acoust., Speech and Sig. Proc.*, 37(1):31–42, January 1989.

[3] M. Effros, P. A. Chou, and R. M. Gray. One-pass adaptive universal vector quantization. In *Proceedings of ICASSP'94*, volume 5, pages 625–628, 1994.

[4] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer, Norwood, MA, 1992.

[5] A. Gersho and M. Yano. Adaptive vector quantization by progressive codevector replacement. In *Proceedings of ICASSP'85*, pages 133–136, 1985.

[6] H. Koga and S. Arimoto. Asymptotic properties of algorithms of data compression with fidelity criterion based on string matching. In *1994 IEEE Int. Symp. Inform. Theory*, page 264, 1994.

[7] A. Ortega and M. Vetterli. Adaptive quantization without side information. In *Proc. ICIP-94*, volume 3, pages 856–860, 1994.

[8] Y. Steinberg and M. Gutman. An algorithm for source coding subject to a fidelity criterion, based on string matching. *IEEE Trans. Inform. Theory*, 39(3):877–886, May 1993.

[9] Z. Zhang and V. K. Wei. An on-line universal lossy data compression algorithm by continuous codebook refinement. In *1994 IEEE Int. Symp. Inform. Theory*, page 262, 1994.