LIKELIHOOD CODEBOOK REORDERING VECTOR QUANTIZATION

Chu Meh Chu (cchu3@gatech.edu), David V. Anderson (anderson@gatech.edu)

School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA, USA

ABSTRACT

In this paper, a reordering vector quantization algorithm based on conditional probabilities of a codebook transition matrix is implemented. The dynamic reordering of the codebook dramatically decreases the entropy for temporally structured sources, enabling a second stage of entropy coding to further improve the efficiency of VQ. Results from synthetic, Markov sources and speech sources are shown to outperform other baseline vector quantization algorithms.

Index Terms— vector quantization, entropy coding, compression

1. INTRODUCTION

Vector quantization (VQ) is a compression technique that is used to encode data such as speech and images for transmission and storage purposes. In contrast with scalar quantization which quantizes on a sample by sample basis the key idea behind vector quantization is to compress a block or frame of samples using a vector codebook. By transmitting only the index of the codebook vector, a better rate/distortion ratio is achieved than is possible with only scalar quantization. Many variations on the basic idea of VQ have been developed and a review of many of these as well as the fundamentals of VQ is presented in [1]. One widely-used VQ training algorithm was developed by Linde, Buzo, and Gray (LBG). The LBG algorithm uses k-means to develop a vector codebook from a set of training vectors using the L_2 distance between the data frames and codebook code vectors as a distortion metric.

While basic VQ methods, such as the LBG algorithm, capture the correlation between the dimensions within a single vector, they do not capture correlation between vectors. For some signals, this vector-to-vector correlation can be significant. However, it is more difficult to capture after quantization because the VQ codebook indices do not reside in a metric space. It is possible to exploit some vector-to-vector correlation using entropy coders such as the LZW coder but for large codebooks, the LZW coder becomes less practical.

Several methods have been proposed to exploit the correlation between vectors. *Address vector quantization* is an adaptive vector quantization algorithm that exploits such correlation between neighboring image blocks. The concept of the address codebook, introduced in [2] uses a codebook that consists of the addresses of the current vector and neighboring horizontal, vertical, and diagonal vectors. This address codebook is re-ordered into active and inactive regions of the codebook using a metric called the score function. The score function is obtained from the probability transition matrix from the neighboring image blocks. The address code vectors with the higher score functions are placed in the $active$ codebook region while the others go to the $inactive$ region.

Cache vector quantization also exploits this temporal or spatial correlation between neighboring input vectors [3]. This method essentially creates a temporary (cache) sub-codebook for each input vector to be encoded by predicting the next vector from the previously encoded input vectors.

A super-codebook is initially designed from which a cache (sub-codebook) is extracted. For every vector to be transmitted, we predict a vector and find a set of code vectors from the super-codebook to form the sub-codebook. The sub-codebook size is usually fixed. To make the codebook size vary with each input vector, a dynamic codebook reordering vector quantization was proposed [4]. This dynamic codebook reordering finds the number of instances when the mean squared error between a code vector in the codebook and the predicted ones is less than mean squared error between the current code vector and the predicted one. This number of instances is the codebook size of the current sub-codebook.

Cache vector and address vector quantization require shuffling the codebook for each input frame. This leads to a constant reordering update of the codebook. In [5], the codebook is reordered once for all input frames according to a similarity metric called the potential function. This metric is essentially the energy of the vector. The potential of a vector is the square of the L_2 norm of that vector. This potential function is used to sort codebook code vectors. This gives it a structure beneficial for differential pulse code modulation (DPCM) to decrease its first order differential entropy which, in turn, increases correlation.

Finite state vector quantization (FSVQ) is another popular approach for doing vector quantization [6]. FSVQ operates by using states that correspond to a set of sub-codebooks. The current state of the encoder is determined by using a next state function. Next state functions use previously encoded blocks and the previous state to predict the current state. Each in-

put frame or vector is vector quantized by searching through the sub-codebook corresponding to a given state. Choosing the appropriate sub-codebooks and the next state function for the FSVQ algorithm are non-trivial problems. Furthermore, FSVQ requires large amounts of memory to handle the various sub-codebooks for each state. These three issues are addressed using the dynamic finite state vector quantization.

Dynamic finite state vector quantization (DFSVQ) is an extension of FSVQ that uses the next state function as a reordering procedure [7]. A super-codebook is initially designed from which a sub-codebook is obtained for each input frame. Encoding each data frame requires reordering of the codevectors in codebook so that the most probable approximations of the input data frame are relocated to the top of the super-codebook. This results in reduction in computational resources and a much simpler determination of the next state of the encoder.

Most of the methods listed above strive to ensure that the majority of VQ indices transmitted come from a smaller set than the full codebook, thus improving coding efficiency.

Dynamic codebook reordering vector quantization (DCR) is also a reordering algorithm that reorders the codebook based on a dissimilarity metric between the quantized current frame and the codevectors in the codebook [8]. This ensures that the indices used are generally near 0 so that subsequent entropy coding can be used to improve overall coding efficiency.

Previous codebook reordering research has focused on reordering the codebook code vectors according to the similarity and dissimilarity metric between the current and previous input quantized data frames. In this paper we introduce the likelihood codebook reordering (LCR) vector quantization algorithm. This algorithm treats the transition between one code vector and the next as a random variable. The basic approach is similar to that used by Krishnan [8] but transition probabilities are explicitly estimated during the training process. A codebook transition matrix is generated by estimating the likelihood of each vector-to-vector transition by counting transitions in the training set. This transition matrix is then used as described below to dynamically reorder the codebook indices in an attempt to cluster the indices used near 0.

In section 2, we describe our algorithm in detail. In section 3, we present and explain our experimental results. Section 4 is comprised of conclusion and recommendations.

2. LIKELIHOOD CODEBOOK REORDERING VECTOR QUANTIZATION

2.1. Conceptual Formulation

For a well-designed vector quantizer, each code vector is equally likely to be chosen; a long-term histogram of the codebook indices generated from the quantization process will be flat. This implies that the entropy of the indices is maximized and no further quantization can be achieved. However, there may be short-term temporal structure that can be exploited. It is well known that real world signals such as speech and images have strong temporarily and spatially correlated regions. Such strong correlations can be exploited so that adjacent code vectors that are $close$ to each other according to some metric may be assigned similar or same symbols in the channel.

In the LCR-VQ algorithm, a count of the transition from one code vector to the other for a given data is computed during training to estimate the transition probability between each code vector pair. Then when quantizing a source, each codebook vector is assigned a new index based on the transition likelihood from the previously used codebook vector. The most likely codebook vector is assigned the index 0 and the next most likely codebook vector is assigned the index 1, and so forth. The result is that, according to the transition likelihoods, low index numbers will be much more common, with 0 being most common—resulting in an easily compressed index bit stream.

2.2. Algorithm

The input data is assumed to be a set of ordered vectors $\{X_t \in \Re^L\}$ originating from some source $\mathbf{S}$ where t is a discrete time index. A codebook, $\mathbf{C} = \{c_0, \ldots, c_{M-1}\}$, is generated using the LBG algorithm, or similar, using a set of training samples from source $\mathbf{S}$. These training samples are then used to estimate the transition matrix T. Each element of T, that is, T_{ij}, represents the likelihood of the index j following index i, or in other words, the likelihood of an input vector, X_{t-1}, that is quantized to codebook vector i being followed by an input vector, X_t, that is quantized to codebook vector j.

Let p represent the previously transmitted index, k is the index of the current frame to be transmitted. Note that different variables other than i and j are used to emphasize the fact that the transmitted or stored index is not the same as the absolute index number of the code vector. Therefore $k = \varphi(i, j, T)$ and not i is the stored or transmitted index.

- Quantize the input sample, X_{t-1}, and note the corresponding absolute codebook index, j.
- Find the set of transition likelihoods from codebook vector j to each code vector in codebook. This corresponds to $T_j = (T_{j1}, T_{j2}, T_{j3}, \ldots, T_{j(M-1)})$.
- Sort the set T_j in descending order. (In practice steps 2 and 3 can be pre-computed or at least partially precomputed.)
- Let $k = \varphi(i, j, T)$ denote the sort order of the i^{th} element of T_j.
- Quantize the next input sample, X_t, and note the corresponding absolute codebook index, i.

- The transmitted or stored index value for X_t is $k = \varphi(i, j, T)$.

Figure 3 gives a simple example of the LCR algorithm. We assume a codebook with 4 code vectors, and a codebook transition matrix as in part a) of Fig. 3 Assuming that the previously quantized frame is code vector 2 (c_2), we want to find how the code vectors (c_1, c_2, c_3, c_4) in the codebook will the reordered in the current frame and transmitted to the communication channel. Part b) of Fig. 3 shows the probability of transmission of the code vectors given the previously transmitted code vector c_2. Part c) shows the reordered codebook in descending order or likelihood to be transmitted. Thus, code vector 2 (c_2) will be transmitted to the channel as index 3 in this case.

a)

Previous \ Next	c1	c2	c3	c4
c1	0.2	0.3	0.4	0.1
c2	0.1	0.2	0.4	0.3
c3	0.4	0.2	0.3	0.1
c4	0.1	0.3	0.2	0.4

b)

Previous \ Next	c1	c2	c3	c4
c2	0.1	0.2	0.4	0.3

c)

Sorted Index	c3	c4	c2	c1

Fig. 3. Example: illustration of the Likelihood Codebook Reordering algorithm.

3. RESULTS

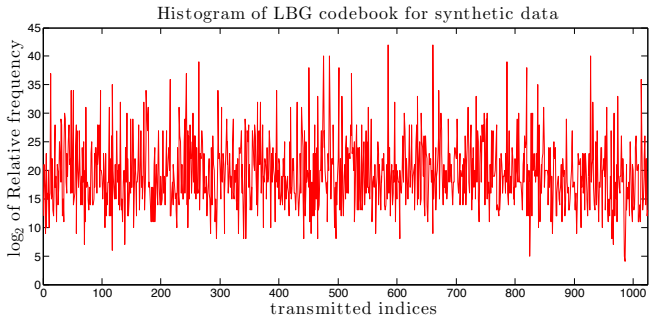

Fig. 1. Histogram of randomly generated data (with correlation = 0.5) quantized using standard LBG vector quantization

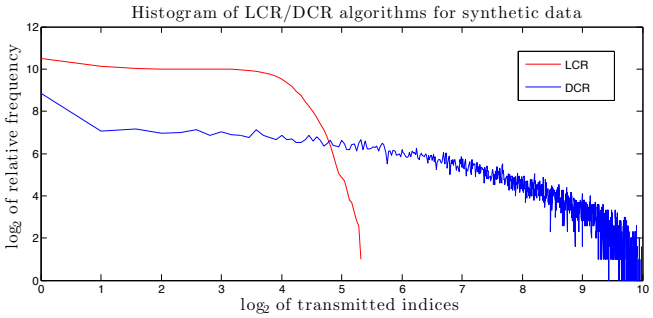

Fig. 2. Histogram of randomly generated data (with correlation = 0.5) quantized by LCR and DCR vector quantization.

We used first-order Gauss-Markov generated data and real TIMIT speech data to test the performance of our LCR algorithm. If $\mathbf{Y}$ is the $d \times L$ data to be quantized, Y assumes the Gauss-Markov property below.

$Y_k = \alpha Y_{k-1} + X_k$
where Y_k is a Gauss-markov vector source, $\alpha \in (-1, 1)$ is the correlation parameter and X_k is an L dimensional vector of an identical independent Gaussian random variable of unit variance. This first-order Gauss-markov model models many real world signals such as speech, video, image, etc. Therefore any results obtained from these synthetic data can be extended to such real applications.

We set L equal to 10 for our experiment and vary the correlation parameter α. A training database of Y of dimension d = 100000 is used while a database of dimension d =20000 is used to test the performance of our LCR algorithm over DCR and LBG.

Once the testing data is LBG quantized, we calculated the codebook transition matrix for the testing data set. The codebook transition matrix is used to calculate the transmitted index. Figure 1 shows the histogram of the testing data set quantized by LBG algorithm. The histogram is very random and has higher entropy than the histogram of both LCR and DCR algorithms. Figure 2 illustrates that both LCR and DCR algorithms reorder the indices to be transmitted so that the lower magnitude indices have higher probability of being transmitted. Figure 2 also highlights the reduced codebook size used by our LCR algorithm compared to both LBG and DCR algorithms. The result of this is lower bit rate for our LCR algorithm.

Real speech samples from the TIMIT database were also used to test the working of our LCR algorithm. Female speech samples from 10 different speakers were obtained from TIMIT database to train the codebook and test our LCR algorithm. Line spectral pairs coefficients were obtained for each speaker sample. The line spectral pairs were obtained by breaking up the speech into frames of 96 samples from which 10 line spectral pairs were obtained. Figure 4 shows the LBG histogram of the speech samples while figure 5 demonstrates that LCR reduces the codebook size relative to that of DCR

and LBG. Bit rate is reduced consequently as a result of this codebook size reduction.

Table 1 and 2 show that the LCR algorithm results in less entropy in comparison to that of LBG and DCR vector quantization. The lower the entropy, the greater the likelihood of a latent structure within the reordered codeboook. Lower entropy is a strong indication of structure and compressability in a communication channel.

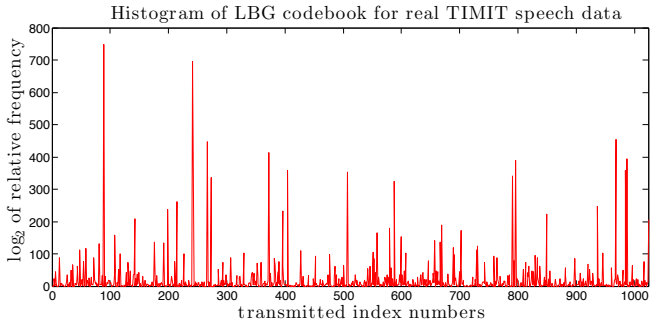

Fig. 4. Histogram of TIMIT speech samples quantized by LBG vector quantization

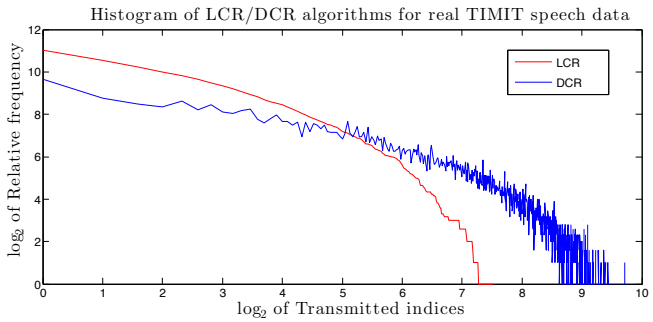

Fig. 5. Histogram of TIMIT speech samples quantized by both DCR and LCR

Correlation	LBG	DCR	**LCR**
0.85	2.9778	2.0430	**1.2697**
0.90	2.9663	1.8288	**1.1681**
0.95	2.9303	1.5019	**0.9880**
0.98	2.8823	1.2927	**0.8573**

Table 1. Table of Entropies verus correlation in Synthetic data comparing LBG, DCR, and LCR algorithms

4. CONCLUSION

We see that our likelihood algorithm reorders the indices to lower ranges. Plots of histograms for both dynamic and likelihood reordering algorithms show a much more reduced codebook size for our algorithm than the DCR. This results in reduced bit rate for either real or synthetic data. In future, we plan to compare our algorithm to other reordering algorithms. We also plan to extend this algorithm to other forms of data such as images, videos, and radar data.

Speech Samples	LBG	DCR	**LCR**
1	2.6855	2.3783	**1.6969**
2	2.4474	2.3675	**1.6799**
3	2.8883	2.3790	**1.6855**
4	2.5772	2.3850	**1.5213**

Table 2. Table of Entropies comparing LCR, DCR, and LBG performance in TIMIT Speech data

5. REFERENCES

[1] R. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4 –29, april 1984.

[2] N.M. Nasrabadi and R.A. King, "Image coding using vector quantization: a review," *IEEE Transactions on Communications,*, vol. 36, no. 8, pp. 957 –971, aug 1988.

[3] F. De Natale, G. Desoli, D. Giusto, C. Regazzoni, and G. Vernazza, "A framework for high-compression coding of color images," pp. 1430–1439, 1989.

[4] F.G.B. De Natale, S. Fioravanti, and D.D. Giusto, "Dcrvq: a new strategy for efficient entropy coding of vector-quantized images," *Communications, IEEE Transactions on*, vol. 44, no. 6, pp. 696 –706, jun 1996.

[5] Guobin Shen and M.L. Liou, "An efficient codebook post-processing technique and a window-based fast-search algorithm for image vector quantization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 6, pp. 990 –997, sep 2000.

[6] J. Foster, R. Gray, and M. Dunham, "Finite-state vector quantization for waveform coding," *Information Theory, IEEE Transactions on*, vol. 31, no. 3, pp. 348 – 359, may 1985.

[7] N.M. Nasrabadi and Y. Feng, "A dynamic finite-state vector quantization scheme," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, apr 1990, pp. 2261 –2264 vol.4.

[8] V Krishnan, *A framework for low bit-rate speech coding in noisy environment*, Ph.D. thesis, march 2005.