Constrained-Storage Vector Quantization with a Universal Codebook *

Sangeeta Ramakrishnan, Kenneth Rose, and Allen Gersho
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106

Abstract

Many compression applications consist of compressing multiple sources with significantly different distributions. In the context of vector quantization (VQ) these sources are typically quantized using separate codebooks. Since memory is limited in most applications, a convenient way to gracefully trade between performance and storage is needed. Earlier work addressed this problem by clustering the multiple sources into a small number of source groups, where each group shares a codebook. As a natural generalization, we propose the design of a size-limited universal codebook consisting of the union of overlapping source codebooks. This framework allows each source codebook to consist of any desired subset of the universal codevectors and provides greater design flexibility which improves the storage-constrained performance. Further advantages of the proposed approach include the fact that no two sources need be encoded at the same rate, and the close relation to universal, adaptive, and classified quantization. Necessary conditions for optimality of the universal codebook and the extracted source codebooks are derived. An iterative descent algorithm is introduced to impose these conditions on the resulting quantizer. Possible applications of the proposed technique are enumerated and its effectiveness is illustrated for coding of images using finite-state vector quantization.

1 Introduction

Vector quantization (VQ) is an appealing coding technique because the rate-distortion bound can be approached by increasing vector dimension, denoted k [1]. In applications where high reproduction quality is required, relatively large values of resolution

*This work was supported in part by the National Science Foundation under grant No. NCR-9314335, and by the University of California MICRO program, Rockwell International Corporation, Echo Speech Corporation, Signal Technology, Inc., Qualcomm, Inc., and Lockheed Missiles and Space Company.

r in bits per sample are needed. This can lead to a very high complexity for such applications, because the computational and storage complexity of unstructured vector quantization grows exponentially with the product kr. In applications where signals from multiple sources are to be encoded, each would normally require separate quantization codebooks, thus further compounding the problem of storage. As a result, codebook storage can become the dominant complexity obstacle.

Constraining the amount of stored data tables in VQ-based coding systems while designing the tables to obtain optimal performance has several applications. In *tree-structured VQ* (TSVQ), the key limitation is storage space since VQ encoding complexity is greatly reduced by the tree structure. In *adaptive VQ* (AVQ), updated codebooks need to be communicated to the receiver periodically. Here, the bit-rate overhead incurred in transmitting codebooks can be greatly reduced by having a fixed "universal" codebook in the receiver from which new codebooks can be specified. If, further, fixed-rate coding is used to specify the new codebook, then the size of the universal codebook determines the rate cost of updating the codebook. Since the storage space of the universal codebook is a key limitation, a constrained storage approach offers an effective solution. In applications where different types of sources need to be encoded, the proposed scheme would be an excellent candidate. For example in image coding, where a wide variety of images like portraits, textures, medical images, text documents etc., need to be encoded. The storage requirements can be reduced by use of constrained storage vector quantization, rather than using individual codebooks for each type of image to be encoded. As a final example, we observe that in *classified VQ* (CVQ) [1] [7] the critical performance limitation is the storage of a large number of class-specific codebooks.

1.1 On the Constrained-Storage VQ Problem

A variety of signal compression applications involving VQ can be restated as follows: Suppose we have a set of N stationary vector sources, each source producing vectors of dimension k, which are to be coded under the same distortion measure and with different resolutions r_i. Generally, a separate codebook for each of the N sources would be required for best performance and each codebook would be designed specifically for one source. To reduce storage requirements, however, codevector sharing is introduced, wherein there exists a universal codebook containing M codevectors and each source uses a particular subset of these vectors as its codebook. A careful choice of codebook, for each source, from the large number of possible codebooks is hence required.

1.2 Relevant Prior Work and Applications

An earlier approach to mitigating the problem of storage complexity in VQ, called *constrained-storage VQ* (CSVQ) was introduced by Chan and Gersho and was based on the concept of *codebook sharing*. This approach directly designs a set of M codebooks and a set of pointers identifying which codebook is to be used by each source, to optimize an overall performance measure for a set of N sources, where $N > M$. The basic theory is given in [2] and a specific design for TSVQ codebooks is given in [3]. In this paper, we shall use the acronym CSVQ to refer more generally to VQ coding schemes that employ storage constraints rather than narrowly to the work of [2].

Another approach suggested for AVQ by Gersho and Gray [1, p. 620] and studied for universal VQ by Zeger et al. [8] is to identify a subset of codevectors from a fixed *universal codebook* that is chosen to represent the current statistics of the source. By effectively designing a suitable universal codebook of limited size, a constrained storage solution to AVQ is obtained.

A third study of CSVQ was reported by Lyons et al. [5]. In this work, the storage space of an optimal large codebook is reduced by quantizing the codevector components with a smaller secondary quantizer of low dimensionality (preferably a scalar quantizer in most cases).

In this work, we examine and solve the optimization problem for a general form of CSVQ which in most respects encompasses the prior work while offering a more versatile solution for several different applications. The basic approach is the optimal sharing of codevectors, rather than sharing of codebooks, while imposing the storage constraint on the total number of codevectors. It provides more efficient storage and allows different codebook sizes for different sources. In particular, it is easy to show that the method of Chan and Gersho [3] is a special case where groups of sources must share the exact same codebook. Our CSVQ will always improve on such a solution by removing these restrictions. The ability to assign different codebook sizes to individual sources makes the method attractive for applications where bit allocation should be used, for example, when the sources correspond to different components or features extracted from a given signal. In such cases, one can optimize the bit allocation, and still impose the practical storage constraint on the overall number of codevectors. This has obvious application in structurally-constrained VQ, where all the codebooks used for the various stages or layers can be extracted from a universal codebook that satisfies the given storage limitations. We note further that this approach can be used in the context of adaptive and universal quantization ([8]), where adaptation is realized by extracting the current codebook from the universal codebook and transmitting the selection parameters to the decoder. In such applications, the size of the universal codebook has direct impact on the rate cost associated with specifying a codebook to the decoder, beside the obvious storage consequences. This approach also finds application in *finite state VQ* (FSVQ) and CVQ, where typically individual codebooks are used for each state/class. By adopting the CSVQ approach being proposed, it is possible to increase the number of states/classes without an increase in memory requirement. Thus, the basic approach proposed here is applicable to a large variety of important applications.

The organization of the paper is as follows. Section 2 discusses CSVQ, gives the formulation of the problem, and provides details about the encoding and decoding rules for our technique. Section 3 presents the design procedure and a brief mention of simulation results that validate this method and Section 4 gives our conclusions.

2 Constrained Storage VQ

2.1 Overview

In many signal compression applications, random vectors from a number of sources need to be encoded. Instead of either using a single common codebook for all the sources (causing a significant drop in performance) or designing separate codebooks for each source (increasing storage requirements), we could adopt a constrained storage vector quantization approach.

Here, we have a single large (universal) codebook, containing codevectors representative of different sources. The codevectors belonging to the universal codebook are referred to as the universal codevectors. Each source uses a portion of the universal codebook as its codebook. Since each codevector is used only by a few sources it is better matched to the source statistics.

In most applications all sources do not need to be quantized at the same rate. CSVQ is very well suited for such applications. Since each source has its own codebook within the universal codebook, we can allow different rates for different sources.

2.2 Formulation

We are given N sources, each generating a k dimensional random vector $\mathbf{X}_n$, for $n = 1, 2, \ldots, N$. A vector quantizer Q_n for the nth source is a mapping of a k dimensional vector in Euclidean space into a member of an ordered finite set C_n of l_n codevectors each of dimension k. The set C_n is the codebook for $\mathbf{X}_n$. To reduce the storage requirements associated with separate source codebooks, we consider a universal codebook S whose size is restricted to $M << \sum_n l_n$ codevectors, i.e., $S = \{\mathbf{s}_i, i = 1, 2, \ldots, M\}$. Of the universal codebook we assign codevectors to each one of the N source codebooks. We denote this assignment by a mapping function $\mu(n)$, where, $\mu(n)$ is a binary vector of dimension M. In particular we write:

$$\mu_i(n) = \begin{cases} 1 & \text{if } \mathbf{s}_i \in C_n \\ 0 & \text{if } \mathbf{s}_i \notin C_n \end{cases}$$

hence, exactly l_n components of $\mu(n)$ are 1. Equivalently, the effective source codebooks are given by:

$$C_n = \{\mathbf{s}_i : \mu_i(n) = 1\}.$$

The constrained storage quantizer is completely specified by the following :

1. The set of M universal codevectors $S = \{\mathbf{s}_i, i = 1, 2, \ldots, M\}$.
2. The mapping $\mu(n)$.

Each vector quantizer, Q_n performs the usual nearest neighbor operation, mapping an input vector $\mathbf{x}$ to the nearest codevector $\hat{\mathbf{x}}$ in its codebook C_n. A codebook C_n is defined to be optimal with respect to the pdf of random vector $\mathbf{X}_n$ if the codebook C_n minimizes the expected distortion $\mathrm{E}d(X_n, Q_n(X_n))$ over all codebooks of size l_n which can be extracted from S.

The performance measure of the multiple source coding system is the overall distortion, and is defined as

$$D = \sum_{n=1}^{N} w_n E(d(X_n, Q_n(X_n))) \tag{1}$$

where w_n are non-negative weights, with $\sum_{n=1}^{N} w_n = 1$, which measure the relative importance of distortion incurred in quantizing the n^{th} source. These weights are application dependent and are assumed to be given prior to the design process. Commonly, the weights are simply the relative frequencies of occurrence of the sources. The set of M universal codevectors and the mapping function $\mu(\cdot)$ are said to be optimal if they jointly minimize the overall distortion D.

2.3 Encoding and Decoding in CSVQ

Suppose a universal codebook with M codevectors $\mathbf{s}_i$ and a mapping function $\mu(\cdot)$ are available at the encoder and decoder. Assume that when a vector $\mathbf{x}$ arrives at the encoder its identification as a member of the n^{th} of the N sources is given. We assume this source label, n, is also available at the decoder.

The encoder identifies the ordered set of codevectors constituting the codebook C_n from the mapping $\mu(n)$ and universal codebook S and performs a nearest neighbor search through this codebook by computing the distortion measure $d(\mathbf{x}, \mathbf{s}_i)$ for each vector in C_n. The input vector $\mathbf{x}$ is thus quantized with a codevector $\hat{\mathbf{x}}$ and the optimal index j^* is transmitted. Assuming fixed-rate coding, $log_2 l_n$ bits are transmitted to specify j^*.

The decoder receives the optimal index j^* and has available the source label n. From its copy of the universal codebook S, the decoder extracts the selected codevector $\hat{\mathbf{x}}$ with the help of the mapping $\mu(n)$ and index j^* and outputs $\hat{\mathbf{x}}$ as the reproduced approximation to the original vector.

Both encoder and decoder need to know to which source the vector being encoded/decoded belongs. In AVQ, the type of source changes infrequently and a number of successive input vectors are treated as emanating from one particular source. This reduces the overhead in transmitting the source label. In certain other applications, the decoder is capable of determining the source index that has been assigned to the encoded vector, due to some prior available information and the source index n need not be explicitly transmitted. For such cases, each successive input vector may be associated with a different source label.

3 CSVQ Design

For a given set of N sources, and a storage restriction of M vectors, we wish to determine the optimal set $S = \{\mathbf{s}_i, i = 1, 2, \ldots, M\}$, of universal codevectors and the optimal mapping function $\mu(\cdot)$. An iterative algorithm can be used, where each step consists of fixing a subset of the parameters while optimizing over the others. This is analogous to known clustering algorithms, such as the Generalized Lloyd Algorithm (GLA) [4].

Starting with an initial universal codebook, and an initial mapping function, we iterate the following two steps :

1. Find the best set of universal codevectors for the given mapping function.
2. Find the best mapping function for the given universal codebook.

Each of these steps is clearly monotonically non-increasing in distortion, so they can be performed iteratively until a fixed point is reached and the distortion converges to some limiting value.

Although the above iteration is conceptually correct, practical considerations may warrant modifications of it in practice. For example, the second step is, in fact, an iterative algorithm in itself, as will be explained below. Thus, it may be advisable to run a single iteration of it (which is still monotonically non-increasing in distortion) and return to step 1, rather than spend computation on determining the absolutely best mapping for an intermediate universal codebook. However, while ignoring issues of speed of convergence, the simplified scheme above does convey the major concepts of the design.

3.1 Universal Codebook

For a fixed mapping function the universal codevectors can be modified to match the sources they represent. The overall distortion for the constrained storage quantizer is given by

$$D = \sum_{n=1}^{N} w_n \sum_{j:\mu_j(n)=1} E[d(\mathbf{X}_n, \mathbf{s}_j)|\mathbf{X}_n \in R_{jn}]P[\mathbf{X}_n \in R_{jn}], \qquad (2)$$

where R_{jn} is the nearest neighbor partition cell corresponding to the codevector $\mathbf{s}_j$ when codebook C_n is used.

For the given mapping function, and nearest neighbor partition R_{jn}, the choice $\mathbf{s}_j$, which minimizes the mean square distortion is simply the centroid, or,

$$\mathbf{s}_j = \frac{1}{(\sum_{n:\mu_j(n)=1} w_n)} \sum_{n:\mu_j(n)=1} w_n E(\mathbf{X}_n|\mathbf{X}_n \in R_{jn}). \qquad (3)$$

Since the centroid minimizes the distortion, the design of the new universal codebook for a given mapping function, does not increase the distortion, i.e., the distortion either decreases or remains the same.

3.2 Mapping Function

Consider the design of C_n given the universal codebook. The problem at hand is that of optimally choosing a subset of l_n codevectors from the set of M universal codevectors. There are obviously too many ($M!/((M-l_n)!l_n!)$) possible choices of mapping, for brute-force exhaustive approaches to be practical. Thus, a simpler, but possibly sub-optimal method is required to determine the mapping. A similar problem has been addressed before in [6]. The heuristic solution proposed there was to determine the relative frequency with which each of the universal codevectors is selected by a particular source, when that source is coded with the entire universal codebook, and then select the l_n most frequently-used universal codevectors as the codevectors for that source.

Here we introduce a new design procedure that ensures convergence to a local optimum. This method can be viewed as a form of the GLA. For a particular source, say source n, we assume we have a training set representing the source statistics. At each iteration of the GLA, each encoding partition, R_{jn}, is updated followed by the updating of the "centroid". However, here we add the necessary constraint that *each of the selected codevectors at every iteration be a member of the universal codebook.* In general one would need to try all universal codevectors and select the one minimizing the distortion. But for the squared error distortion, as we shall show below, we can equivalently compute the unconstrained centroid, and then quantize it using the universal codebook.

The following three steps are performed at each step of the iteration :

1. Perform a nearest neighbor search using the old codebook.
2. Compute the centroid for each partition.
3. Quantize each of the centroids using the universal codebook.

We need to show that the iterations yield monotonically non-increasing distortion. It is well known from GLA that step 1 cannot increase the distortion. We now show that steps 2 and 3 together will not increase the distortion either. In fact, what we demonstrate is that steps 2 and 3 are equivalent to selecting the best universal codevector for the given partition cell. We need to find $\mathbf{s}$ in the universal codebook to minimize

$$D_{jn} = E[(\mathbf{X}_n - \mathbf{s})^2 | \mathbf{X}_n \in R_{jn}] = E[(\mathbf{X}_n - \mathbf{m}_{jn})^2 | \mathbf{X}_n \in R_{jn}] + (\mathbf{m}_{jn} - \mathbf{s})^2, \qquad (4)$$

where $\mathbf{m}_{jn} = E[(\mathbf{X}_n | \mathbf{X}_n \in R_{jn}]$ is the computed centroid of the cell R_{jn}. Clearly, finding the optimal codevector in the universal codebook is equivalent to quantizing m_{jn} with the nearest universal codevector. Switching from the current universal codevector to the best universal codevector can never increase the distortion.

To summarize, the basic iteration is guaranteed not to increase the distortion, hence it will decrease the distortion until convergence. This procedure can be performed on each source independently. At convergence the algorithm produces an (at least locally) optimal mapping and encoding partition for the given fixed universal codebook.

3.3 Practical Design Algorithm

Similar to the GLA, the CSVQ method is applicable to sources given by probability distributions, and to sources represented by training sets. In the latter case, one simply assumes that the source distribution is well approximated by the discrete distribution induced by a uniform probability over the samples of the training set.

Thus all our results can be restated within the context of a training algorithm in a straightforward manner.

In the following sketch we summarize the practical algorithm that we used in our simulations.

Given a training set for each source, compute an initial universal codebook $S^{(0)}$ by applying GLA to the entire training data. Compute initial codebooks $C_n^{(0)}$ by applying GLA to each training set separately. Initialize $m = 1$.

1. Update the mapping:
   (a) Do Nearest Neighbor encoding of the training set vectors for each source using the corresponding codebook ${C_n}^{(m-1)}$.
   (b) Compute centroids for every partition of each source.
   (c) Quantize these centroids using the universal codebook, thus obtaining ${C_n}^{(m)}$. This determines the mapping $\mu^{(m)}(n)$.
2. Update the universal codebook : The training set for each source is encoded using $\mu^{(m)}(n)$ to determine the nearest neighbor partition cells $R_{jn}^{(m)}$. Then each universal codevector $\mathbf{s}_j^{(m)}$ is obtained by (3).
3. Check convergence. If not converged, increment m. Go to 1.
4. STOP

3.4 Experimental Results

The proposed technique was applied to finite state vector quantization (FSVQ) of images. At a given rate, FSVQ's performance can be improved by increasing the number of states. However, since a codebook is assigned to each state, this results in increasing memory requirements. Thus, in order to optimize the overall performance subject to the specified rate and available memory, we apply our constrained storage approach, and the resulting algorithm is referred to as constrained storage FSVQ (CS-FSVQ). In particular, by allowing codevector sharing we can increase the number of states while respecting the given memory constraint.

For the simulations we selected a data set of ten images of different types. We segmented them into 4 x 4 blocks to create the training set. Given a fixed rate, and based on the training set we applied the omniscient design method [9] to design a sequence of five standard FSVQ systems with number of states $N = 4, 8, 16, 32, 64$, respectively. All codebooks in all systems were of the same size as determined by the fixed rate.

The next state function was implemented as a vector quantizer operating on the pixels in the 3 neighbouring blocks as shown in Figure 1. In Figure 2 we show the SNR versus memory for these systems. We then used the standard FSVQ points for $N = 16, 32, 64$ as initialization and constrained the memory size to obtain the curves depicted in Figure 2. More precisely, we used the union of codebooks of a given

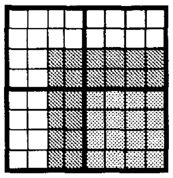

Figure 1: Current state is determined from the neighboring pixels in the previously encoded blocks.

standard FSVQ solution as training set to extract (using GLA) an initial universal codebook of the restricted size M. This provided the initialization for the algorithm described in Section 3.3.

The results demonstrate that the CS-FSVQ approach improves over standard FSVQ in two respects. First it provides solutions at arbitrary levels of memory, while standard FSVQ can only provide systems at distinct memory size corresponding to the integer number of states. Secondly, and more importantly, CS-FSVQ consistently outperforms standard FSVQ at all memory sizes. Please note that the memory values used in Figure 2 were adjusted to include the memory cost of implementing the next-state function. This gives a fair comparison and penalizes our CS-FSVQ results that allow more states. However, other forms of next-state function (instead of the VQ we used) may require less memory.

4 Conclusion

We have introduced a new codevector sharing approach to Constrained Storage VQ, for quantizing multiple sources. The method uses a single universal codebook whose size is limited by the specific application. A number of sources may use the same universal codevector, but they are not constrained to use exactly the same set of codevectors for their codebooks. This technique trades off performance for large reductions in storage complexity, and provides an (at least locally) optimal design for a given memory size. The technique is a fundamental one in the sense that it is directly applicable to many different problems including a variety of structured VQ schemes, such as FSVQ, CVQ, and TSVQ, and various problems of universal quantization.

References

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression* , Kluwer Academic Publishers, 1991.

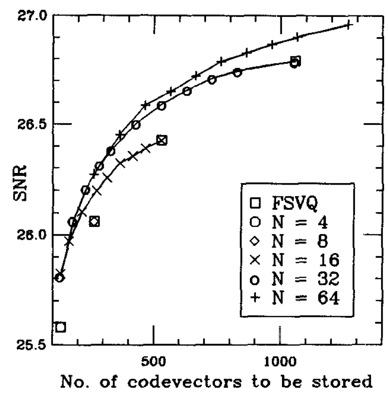

Figure 2: Comparison of CS-FSVQ and standard FSVQ. The plot shows SNR versus memory size, each curve corresponds to a different number of states. The rate is 0.3125 bits/pixel.

[2] W. Y. Chan and A. Gersho, "Constrained-storage quantization of multiple vector sources by codebook sharing," *IEEE Trans. on Commun.*, Vol. 39, No. 1, pp. 11–13, Jan., 1991.

[3] W. Y. Chan and A. Gersho, "Constrained-Storage Vector Quantization in High Fidelity Audio Transform Coding," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc.*, Toronto, Canada, pp. 2497-3600, May 1991,

[4] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, Vol. COM - 28, pp. 84–95, Jan. 1980.

[5] D.F. Lyons, D.L. Neuhoff, and D. Hui, "Reduced Storage Tree-Structured Vector Quantization," *Proc. IEEE Conf. Acoustics, Speech, Signal Proc.,* Minneapolis, vol. 5, pp. 602-605, April 1993.

[6] N. M. Nasrabadi, C. Y. Choo and Y. Feng, "Dynamic finite-state vector quantization of digital images," *IEEE Trans. Commun.*, Vol. 42, No. 5, pp. 2145–54, May 1994.

[7] B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Trans. Commun.*, Vol. COM - 34, No. 11, pp. 1105–15, Nov., 1986.

[8] K. Zeger, A. Bist, and T. Linder, "Universal source coding with codebook transmission," *IEEE Trans. on Commun.* , vol. 42, pp. 336-346. Feb./Mar./Apr. 1994.

[9] J. Foster, R. M. Gray, and M. Ostendorf Dunham, "Finite-state vector quantization for waveform coding," *IEEE Trans. Inform. Theory* , IT-31:348-359, May, 1985.