A PARALLEL PROCESSING ALGORITHM FOR VECTOR QUANTIZER DESIGN BASED ON SUBPARTITIONING

Kenneth Zeger † and Allen Gersho ††

†Department of Electrical Engineering
Holmes Hall 483
University of Hawaii
Honolulu, HI 96822

††Center for Information Processing Research
Department of Electrical & Computer Engineering
University of California,
Santa Barbara, CA 93106

Abstract

A technique is presented for designing vector quantizers that is well suited for parallel processing environments. The input space is iteratively partitioning into M disjoint connected regions composed of unions of partition regions. Each of N processors then independently computes an optimal "subquantizer" for its restricted input space. The partitions can regularly be changed to further improve the overall quantizer performance. An additional benefit of this technique is that the the technique can improve on the performance of the generalized Lloyd algorithm by following the traditional design process with the subpartitioning iterations.

1. Introduction

The generalized Lloyd algorithm (GLA) provides a useful technique for designing vector quantizers (VQ) from a finite training set of input data. The GLA is an iterative descent technique which converges to a locally optimal solution in a finite number of steps. Two noticeable disadvantages of the GLA are its computational complexity and its inability to find globally optimal solutions. The computational complexity of the GLA grows linearly with the number of codevectors and the size of the training set, and hence can become burdensome or unusable when very large training sets are needed to adequately represent the source. Also, the locally optimal codebooks produced by the GLA can be substantially inferior to those obtainable by other even more complex optimization techniques, such as simulated annealing. For these reasons, it is valuable to investigate quantizer design algorithms that yield performance equivalent to or better than the GLA but with reduced execution times. Furthermore, algorithms which are amenable to parallel processing are of great interest for future parallel computation environments.

In this paper we present a new quantizer design technique, called *subpartition vector quantization*, which is highly applicable to parallel processing, and achieves equal or better performance than the GLA. The basic idea is to partition the input space into M disjoint connected *subpartition* regions that are the union (not necessarily convex) of current partition regions and redesign separate *subquantizers* for the restriction of the source to each subpartition region. That is, each subquantizer is a quantizer whose input space is one of the M subpartition regions. Each of the M subpartition regions is a union of some of the nearest neighbor regions induced by the codebook. A unique processor is assigned the task of designing a subquantizer for each subpartition region. This process is then iterated by repartitioning the entire input space. The result is that good quantizers based on very large amounts of training data can be developed efficiently using many processors simultaneously.

It should be noted that even if only a single processor is available, a reduction in computational complexity can still be achieved since repetition of many small quantizer design tasks is often more efficient that performing one large quantizer design.

2. Subpartitioning the Input Space

Consider a vector quantizer Q that maps k-dimensional Euclidean space $\mathbf{R}^k$ into a codebook $C = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$, containing N vectors from $\mathbf{R}^k$. Associated with each codevector $\mathbf{y}_i$ is a partition region R_i and a probability p_i, such that for every vector $\mathbf{x}$ in R_i, $Q(\mathbf{x}) = \mathbf{y}_i$, and p_i is the probability of region R_i. A *subpartition* of the partition $\{R_i : 1 \le i \le N\}$ is determined by a partition of the index set $\{1, 2, \cdots, N\}$ into M mutually disjoint sets Λ_i for $1 \le i \le M$. Subpartition regions are defined to be the M unions, $\hat{R}_j = \bigcup_{i \in \Lambda_j} R_i$, as illustrated in Figure 1.

Corresponding to each region $\hat{R}_j$ is a *subquantizer* Q_j whose codebook C_j is the set of codevectors from the codebook C with indexes in Λ_j. Thus, C_j contains exactly one codevector for each index in Λ_j. The average distortion of the quantizer Q can be written

$$D = E[d(\mathbf{x}, Q(\mathbf{x}))] = \sum_{i=1}^{N} D_i$$

where for each i,

$$D_i = p_i E[d(\mathbf{x}, \mathbf{y}_i) \mid \mathbf{x} \in R_i]$$

is the partial distortion associated with region R_i. In terms of the subpartition regions we can write

This work was supported by the University of California MICRO program, Bell Communications Research, Bell-Northern Research, and Rockwell International and by the National Science Foundation.

CH2977-7/91/0000-1141 $1.00 © 1991 IEEE

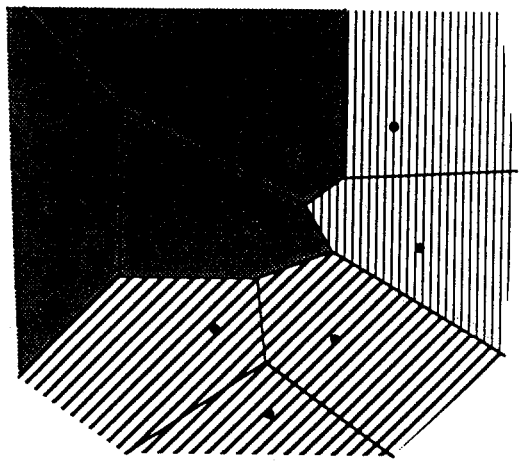

Figure 1: Example of a subpartition of two-dimensional space. The quantizer has N=8 codevectors and M=3 processors are available. Each processor operates on a distinctly shaded union of Voronoi cells.

$$D = \sum_{i=1}^{M} \sum_{j \in \Lambda_i} D_j.$$

The distortion of a particular subpartition quantizer is

$$D_{\Lambda_i} = \frac{\sum_{j \in \Lambda_i} D_j}{q_i}$$

where $q_i = \sum_{j \in \Lambda_i} p_j$ is the probability of the subpartition region $\hat{R}_j$. Thus, the the distortion can be expressed in the form:

$$D = \sum_{i=1}^{M} q_i D_{\Lambda_i}.$$

For a given collection of index sets $\{\Lambda_j : 1 \le j \le M\}$, it is clear that by reducing the distortion D_{Λ_i} of any subpartition quantizer, the overall quantization distortion D will be reduced as well. This provides the basis for a useful VQ parallel design strategy that focuses on optimally designing M independent quantizers $Q_1, \cdots, Q_M$. The input space of the i^{th} quantizer Q_i is assumed to be the set $\hat{R}_i$. The idea is to have every vector quantizer Q_i concurrently designed by one of M independent processors.

3. Parallel Design

There are two main issues that arise with this scheme. One is the question of how to divide the input space $\mathbf{R}^k$ into M disjoint regions and the other issue is how to design each of the M independent subpartition quantizers once the input space is partitioned.

Given M available processors, one would like to partition $\mathbf{R}^k$ into M regions such that the distortion of each subquantizer can be substantially reduced. That is, an iterative design algorithm begins with a collection of N codevectors and a nearest neighbor partition of $\mathbf{R}^k$. That is, $\mathbf{R}^k$ is divided into a disjoint union of N convex polytopal regions R_i. Then a subpartition is obtained after which M subquantizers are designed in parallel. The performance of the overall quantizer Q is improved if at least one of the subquantizer's performance is improved.

Intuitively, the performance of a subquantizer, Q_i, can be improved more easily if its input space, $\hat{R}_i$, is connected. It is well known that a necessary condition for quantizer optimality (assuming a squared-error distortion criterion) is that each codevector be the centroid of its associated partition region. If this centroid condition is satisfied by the quantizer Q prior to subpartitioning (as might be expected), then improvement in one of the subquantizers can occur only by repartitioning the input space of the subquantizer; e.g. by using the nearest neighbor rule in the GLA for the subquantizer. On one extreme, if the input space $\hat{R}_i$ of quantizer Q_i is "completely disconnected" (in the sense that for every pair of distinct $j, k \in \Lambda_i$, R_j and R_k are not neighbors), then Q_i very likely already satisfies the nearest neighbor condition, so that no improvement can occur for Q_i. One thus strives for the other extreme, that each input space $\hat{R}_i$ be connected; i.e. for every $j \in \Lambda_i$, there exists a $k \in \Lambda_i$ such that R_i and R_k are neighbors.

Another constraint that helps to maximize throughput is that each of the M processors (assumed for simplicity to be equivalent) be given an equal amount of computational tasks to perform. This can roughly be achieved by requiring that all subpartition index sets Λ_i be of equal size, i.e. N/M. It may sometimes be convenient to assume this constraint, and also to assume that N and M are integral powers of two, though these are certainly not necessary.

While there are many effective methods for constructing the sets Λ_i, the complexity of this construction must be kept small so as not to override the benefit of complexity reduction achieved by parallel processing. One such technique for clustering the regions R_i into M collections of regions involves executing a "mini-GLA", using the N codevectors of Q as a training set to design a "codebook" of size M. The resulting "partition" will naturally cluster the N codevectors of Q (and hence the regions R_i) into M connected groups which thus define the sets Λ_i (of nearly equal size usually). The codevectors in each of the M subpartition regions are used to partition the input training set of Q into M individual "sub-training" sets, one for each subpartition region.

For example, if Q has a codebook of size 1024 and 16 processors are available, then the "mini-GLA" first designs a locally optimal codebook of size 16 for Q. The resulting 16 partition regions determined by this codebook are used to cluster the 1024 codevectors of Q into 16 groups, each containing on average 64 codevectors. The original training set is then divided into 16 subsets, one for each available processor.

One important special case is that of "two-point" subquantizers; i.e. each subquantizer contains exactly two codevectors. The design of two-point quantizers can be made quite efficient and is used as the basis of many tree-structured vector quantizer design algorithms that employ the "splitting" method [1]. The design of optimal two-point scalar quantizers has recently been studied by [2]. To partition the input space into pairs of neighboring regions, one can either use the mini-GLA technique described previously or else one can use available algorithms from computational

geometry for finding neighboring cells in multidimensional Voronoi diagrams [3].

4. Subquantizer Design and Complexity

Once the subpartitions are determined, the subquantizers must be designed. The complexity of designing each subquantizer is substantially reduced from that of designing the quantizer Q alone. The input space of each subquantizer is a subset of the entire space $\mathbf{R}^k$. The size of the training set for each subquantizer is on average one M^{th} of the size of Q's training set and the size of each subquantizer's codebook is about one M^{th} the size of $Q's$ codebook. The computational complexity of each iteration of the GLA is linearly proportional to the size , N, of the codebook and to the size, T, of the training set; that is $O(NT)$. The computational complexity of running the GLA for a particular subquantizer is thus seen to be

$$O(\frac{N}{M}\frac{T}{M}) = \frac{1}{M^2}O(NT),$$

so that the complexity if reduced by a factor of M^2. Here we assume that the number of iterations in each GLA remains approximately constant, which in practice is a reasonable assumption.

Thus if only one processor is available, a computation savings of about a factor of M is achieved, since M subquantizer designs are required. With M available processors, the computational savings factor is about M^2, which is quite substantial. Even if multiple iterations of subpartitioning are used, a large computational saving can still be achieved.

Since the sizes of the training sets and codebooks for the subquantizers are reduced, more complicated design algorithms can be used to obtain better performance in the subquantizers than using the GLA. For example, the best performance of repeated uses of the GLA with different initialization conditions can be used, as can various other techniques such as stochastic relaxation [4] and Kohonen learning [5], which can yield higher performance codebooks at the expense of greater complexity.

Following the completion of all the subquantizer designs, a single iteration of the GLA on the entire quantizer Q can be done to further improve things. The motivation of this technique is to spend a relatively large percentage of the computational effort performing subquantizer design (with many processors acting simultaneously) and a small amount of time optimizing the original quantizer.

In summary, the total complexity of the proposed scheme (without any GLA iterations on the entire quantizer Q) consists of the computation of designing the subquantizers and of determining a repartition. Determining the repartition can be done very quickly using the "mini-GLA" technique described previously and the subquantizer design can be done at a greatly reduced complexity than the GLA on the whole quantizer.

5. Algorithm

A brief description of the algorithm is summarized below.

1) Choose an initial codebook C_1 of size N.
2) Design a codebook C_2 of size M.
3) Partition C_1 into M sets according the nearest neighbor regions of C_2.
4) Design M subquantizers according to the nearest neighbor subpartition regions determined in Step 3.
5) Goto Step 2.

6. Conclusion

The proposed scheme can provide a significant reduction in complexity over traditional VQ design techniques such as the GLA. In addition to the computational savings using a parallel machine, it should also be noted that actual gain in the system's signal-to-noise ratio can also be achieved by first running the GLA until it converges and then performing the subpartition quantizer improvements. The locally minimal state that results from the GLA can be improved by replacing some of its locally optimal subpartition quantizers with global ones. This then allows further improvement via Lloyd iterations since the codevectors have been perturbed.

7. References

[1] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design", *IEEE Trans. Communications*, COM-28, 84-95, January 1980.

[2] X. Wu, "Optimal Bi-Level Quantization and Its Application to Multilevel Quantization", *IEEE Trans. on Information Theory*, IT-37, January 1991.

[3] F.P. Preparata and M.I. Shamos, "Computational Geometry: An Introduction", Springer-Verlag, New York, 1985.

[4] K. Zeger and A. Gersho, "A Stochastic Relaxation Algorithm for Improved Vector Quantiser Design", *Electronics Letters*, Vol. 25, No. 14, pp. 896-898, July 1989.

[5] T. Kohonen, *Self Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.