A New Initialization Technique for Generalized Lloyd Iteration

Ioannis Katsavounidis, C.-C. Jay Kuo, and Zhen Zhang

Abstract—**The generalized Lloyd algorithm plays an important role in the design of vector quantizers (VQ) and in feature clustering for pattern recognition. In the VQ context, this algorithm provides a procedure to iteratively improve a codebook and results in a local minimum that minimizes the average distortion function. In this research, we propose an efficient method to obtain a good initial codebook that can accelerate the convergence of the generalized Lloyd algorithm and achieve a better local minimum as well.**

I. Introduction

A VECTOR quantizer Q of dimension k and size N is a mapping from a vector in k-dimensional Euclidean space R^k into a finite set C containing N output or reproduction points. We can write it mathematically as Q: $R^k \rightarrow C$, where $C = \{y_1, y_2, \ldots, y_N\}$, and $y_i \in R^k$. The set C is called the codebook, and y_i, $1 \leq i \leq N$, are called the code vectors or codewords. To measure the vector quantizer performance, a distortion measure $d(x, Q(x))$ has to be defined in association with any input vector x and its reproduction vector $Q(x)$. With such a measure, one can quantify the performance of a vector quantizer by either the average distortion $D = E[d(x, Q(x))]$ or the worst-case distortion $D_{\max} = \max_x d(x, Q(x))$. To permit tractable analysis and easy evaluation, the distortion measure is often chosen to be the squared error $d(x, Q(x)) = ||x - Q(x)||^2$.

The design of an optimal quantizer is to seek the codebook that minimizes the average distortion over all possible codebooks. It can be easily shown that the optimal quantizer must satisfy the following two conditions. First, it must be a nearest neighbor quantizer, i.e., it assigns to an arbitrary vector the codeword that is closest to it. Second, for a given partition of the feature space, it must satisfy the centroid condition, i.e., each codeword must be the centroid of the vectors that are mapped to it. The above two optimality conditions provide an algorithm for the design of a locally optimal codebook with iterative codebook improvement. This algorithm is known as the generalized Lloyd iteration where each iteration consists of the following two steps:

1. Given a codebook $C_m = \{y_i; i = 1, \ldots, N\}$ obtained from the mth iteration, find the optimal partitioning of the space R^k, that is, use the nearest-neighbor condition to form the nearest-neighbor cells $R_i = \{x\colon d(x, y_i) < d(x, y_j); j \neq i\}$.
2. Use the centroid condition to update the codebook $C_{m+1} = \{\text{centroid}(R_i); i = 1, \ldots, N\}$, which is the optimal reproduction codebook based on the cells found in (1).

The above algorithm is an iterative optimization procedure based on the method of alternating projections and, therefore, leads to a local minimum.

It has been observed that both the convergence rate of the generalized Lloyd iteration and the performance of the converged codebook depend on the initial codebook C_0. Thus, it is important to find a good initial codebook. Many different initialization methods have been proposed, including random coding, pruning, pairwise nearest-neighbor design, product code, and splitting. A thorough survey of these methods can be found in [2]. Although random coding and product codes are easy to implement, they provide poor results, i.e., a larger number of generalized Lloyd iterations and a poor local minimum as the ultimate codebook. The pruning, pairwise nearest-neighbor design and splitting initialization methods give better results but have a higher computational complexity. Equitz [1] proposed a reduced-complexity pairwise nearest-neighbor method that produced better results than random initialization.

To improve the performance of the VQ codebook, methods other than the standard GLA have been examined by researchers. Yair *et al.* [4] suggested an on-line (or serial) version of the GLA that achieved better results than some implementations of the GLA. Zeger *et al.* [5] proposed to use stochastic relaxation, such as the simulated annealing method, to achieve a better result than GLA. These methods show an average improvement of 0.3 dB in codebook performance, but they require significantly more execution time than the GLA.

In this letter, we propose a new simple initialization technique that leads to a fast convergence behavior and a converged codebook with an excellent performance in terms of average or worst-case distortion. Note that the work in [4] and [5] addressed the optimality of the GLA and proposed substitutes for that while we are trying to find a way to improve the result by changing the initial conditions of the GLA.

Manuscript received March 14, 1994; approved July 15, 1994. This work was supported by the National Science Foundation Presidential Faculty Fellow Award ASC-9350309.

I. Katsavounidis and C.-C. J. Kuo are with the Signal and Image Processing Institute and the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-2564 USA.

Z. Zhang is with the Communication Sciences Institute and the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-2565 USA.

IEEE Log Number 9405628.

II. New Initialization Technique

The idea behind our technique is similar to that of pruning, that is, we pay attention to the training vectors that are most far apart from each other because they are more likely to belong to different classes. Let v_i, $i = 1, \ldots, M$ be the training sequence of vectors. The procedure can be stated as follows:

1. Calculate the norms of all vectors in the training set. Choose the vector with the maximum norm as the first codeword.
2. Calculate the distance of all training vectors from the first codeword, and choose the vector with the largest distance as the second codeword. Then, we have a codebook of size 2.
3. Generally, with a codebook of size i, $i = 2, 3, \ldots$, we compute the distance between any remaining training vector v_k and all existing codewords and call the smallest value the distance between v_k and the codebook. Then, the training vector with the largest distance from the codebook is chosen to be the $(i+1)$th codeword. The procedure stops when we obtain a codebook of size N.

The essence of the above procedure is to use the vector that is most different from existing code vectors as the new codeword. Note that in Step 3, we only need one distance computation for each training vector at each iteration since only one new member is added to the codebook. Based on this observation, it is easy to see that it takes N iterations to obtain a codebook of size N, and each iteration requires M distance calculations, and consequently, the entire initialization procedure has a complexity of $O(MN)$. This is exactly the same as the complexity of one generalized Lloyd iteration.

The proposed initialization scheme has several attractive properties. First, it can be applied to arbitrary codebook sizes (not only of integer resolution as is the case for product codes and splitting). Second, there is no need for the specification of a threshold, like the pruning method. Third, with proper implementation, we can keep the minimum distance as well as the associated codeword for each training vector during the initialization phase. Thus, we can also obtain the initial partition as the byproduct. This means that by doing the proposed initialization procedure, we are in fact performing the first generalized Lloyd iteration.

III. Performance Evaluation

We find it difficult to provide a performance analysis of the proposed new method and will instead present some experimental results to illustrate its performance. We applied the new method to a set of training sequences obtained from three test images. They are the well-known baboon, Lena, and boat images from the USC image database, which have a size of 512×512 pixels and 256 gray scales per pixel. We examined different block sizes including 2×2, 4×4, and 8×8.

Our method was compared with the splitting method [3], which is, in fact, the primary competitor. We restricted our experiments to integer resolution, i.e., a codebook of size $N = 2^m$ for some integer m. However, it must be emphasized that this is a restriction of the splitting and *not* of our method. We allowed two generalized Lloyd iterations in the intermediate stages of the splitting process. Since the complexity of one generalized Lloyd iteration at resolution 2^i is $2^i M$, the complexity of the whole initialization process is $2\sum_{k=0}^{m-1} 2^k M = 2M(2^m - 1)$, which is equivalent to the computational complexity of two generalized Lloyd iterations at the finest (i.e., 2^m) level.

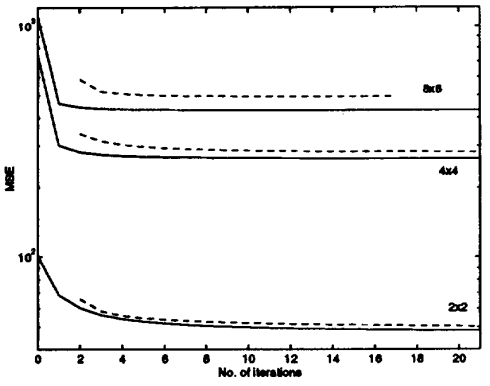

Fig. 1. Convergence history of generalized Lloyd iteration with the splitting (dashed) and proposed new (solid) initialization methods.

TABLE I
MSE Values of Codebooks Obtained in 20 Generalized Lloyd Iterations

	4 x 4 blocks			8 x 8 blocks		
N	Split.	New	Gain	Split.	New	Gain
128	375.835	361.250	0.17	641.853	614.727	0.19
256	329.637	311.371	0.25	581.776	533.846	0.37
512	283.046	265.045	0.29	492.881	430.460	0.59
1024	238.222	217.555	0.39	347.855	278.661	0.96
2048	188.564	163.055	0.63	135.595	96.667	1.40

We report the results for the baboon image in Fig. 1 and Table I. In Fig. 1, we plot (in semi-logarithmic scale) the MSE value versus the number of generalized Lloyd iteration with different block sizes by choosing $N = 512$. Recall that the initialization with the splitting method is equivalent to two generalized Lloyd iterations while our method does not add extra cost. Thus, to make a fair comparison, the solid curve (our method) starts with 0 iteration, whereas the dashed curve (splitting) starts with two iterations. We see that the MSE decays very quickly (in one iteration) for our method and has a lower converged value. We also list the MSE values for various codebook and block sizes after 20 generalized Lloyd iterations in Table I. We see that our method provides a better codebook with lower MSE values for all cases. To characterize the gain, we computed the ratios of MSE values of two methods and expressed them in decibels. When the vector dimension k or the codebook size N becomes larger, the gain is more pronounced. Generally speaking, the improvement is very impressive. The same results were also observed for the Lena and boat test images.

IV. Conclusion

An efficient initialization technique for the generalized Lloyd iteration was proposed in this letter. It reduces the computational complexity and achieves a better local minimum.

Thus, it provides an excellent choice for the implementation of the generalized Lloyd algorithm in both vector quantization and unsupervised clustering applications.

REFERENCES

[1] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 37, no. 10, pp. 1568–1575, Oct. 1989.

[2] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston: Kluwer, 1992.

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

[4] E. Yair, K. Zeger, and A. Gersho, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 40, no. 2, pp. 294–309, Feb. 1992.

[5] K. Zeger, J. Vaisey, and A. Gersho, "Globally optimal vector quantizer design by stochastic relaxation," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 40, no. 2, pp. 310–322, Feb. 1992.