VECTOR QUANTIZATION CLUSTERING USING LATTICE GROWING SEARCH

Dorin Comaniciu *
CAIP Center, Rutgers University
Frelinghuysen Rd, Piscataway, NJ 08855, USA
Tel: (908) 445-0549; fax: (908) 445-4775
e-mail: comanici@caip.rutgers.edu

Cristina Comaniciu
Dept. of Applied Electronics
Polytechnic University of Bucharest
313 Spl. Independ., 77206 Bucharest, Romania
e-mail: ccoman@pcnet.pcnet.ro

ABSTRACT

In this paper we introduce a non-iterative algorithm for vector quantization clustering based on the efficient search for the two clusters whose merging gives the minimum distortion increase. The search is performed within the K-dimensional cells of a lattice having a generating matrix that changes from one step of the algorithm to another. The generating matrix is modified gradually so that the lattice cells grow in volume, allowing the search of the two closest clusters in an enlarged neighborhood. We call this algorithm *Lattice Growing Search* (LGS) clustering. Preliminary results on 512×512 images encoded at 0.5 bits/pixel showed that the LGS technique can produce codebooks of similar quality in less than 1/10 of the time required by the LBG algorithm [9].

1 INTRODUCTION

The standard method for obtaining a codebook is the *Generalized Lloyd* or *LBG* algorithm [9], which monotonically decreases the distortion function towards a local minimum. Although conceptually simple and easy to implement, the LBG algorithm is not computationally efficient due to its exhaustive checking of every training vector against every codeword for the closest match. This complexity of LBG and the appearance of Vector Quantization (VQ) [6] methods that require on-line codebook generation, such as adaptive VQ with codebook transmission [13] or vector quantization with training set synthesis [3], have induced an increased research on faster clustering techniques.

Among the various efficient algorithms recently proposed in literature we note the fast PNN clustering of Equitz [5] (PNN is actually identical to Ward's hierarchical clustering [11, 4]), Directed-Search Binary Splitting [7], Maximum Descent [1], Acceleration of LBG [12], Mean Pyramid of Codewords [8], and Modified PNN [2]. These methods usually try to identify a particular input data structure for a faster search and/or to exploit some general descriptors of the input data to eliminate redundant searches.

The basic idea of the Lattice Growing Search (LGS) algorithm that we introduce here is similar to that of hierarchical clustering. We start with a separate cluster for each training vector and continue merging the closest clusters (two clusters at a time), until the desired codebook is obtained. To efficiently find the cluster pair for merging, we use a growing lattice in the K-dimensional space with the purpose of a pre-classification of current clusters. This pre-classification considerably reduces the number of distance calculations required to detect the optimal merging, without sacrificing performance.

Section 2 of this paper presents the main features of the proposed LGS algorithm. In Section 3 the results of computer simulations are given. Section 4 contains some concluding remarks.

2 LGS CLUSTERING

2.1 Optimum Merging in Hierarchical Clustering

We assume that the distortion measure in evaluating the clustering quality is the squared error. The balance of mean squared errors that corresponds to the merging of clusters C_i and C_j into the cluster $C_{i,j}$ is given by [5]

$$n_{ij}D_{ij} = n_iD_i + n_jD_j + \Delta_{ij} \tag{1}$$

where we use the following notation:

- n_i is the number of training vectors in C_i, n_j corresponds to C_j, and $n_{ij} = n_i + n_j$ represents the number of training vectors in $C_{i,j}$;
- $D_i = (1/n_i) \sum_{\underline{X} \in C_i} \|\underline{X} - \underline{\nu}_i\|^2$ is the mean squared error between $\underline{\nu}_i$ and the training vectors $\underline{X}$ in C_i; D_j and D_{ij} are defined in the same manner as D_i;
- $\underline{\nu}_i$, $\underline{\nu}_j$, and $\underline{\nu}_{ij}$ denote the centroid of training vectors in C_i, C_j, and C_{ij}, respectively;
- $\Delta_{ij} = (n_in_j/(n_i + n_j)) \|\underline{\nu}_i - \underline{\nu}_j\|^2$ is the squared error introduced by merging the clusters C_i and C_j.

The optimum merging is achieved when the quantity Δ_{ij} is minimized over all possible cluster pairs (C_i, C_j). Clearly, the minimization of Δ_{ij} is a computationally

*Part of this work was done while the first author was with the Dept. of Applied Electronics, Polyt. Univ. of Bucharest

expensive task, due to the large number of initial clusters (in the beginning, the number of clusters is equal to the number of training vectors, which is generally large).

However, within the first steps of the clustering process, the number of vectors in each cluster is small. Therefore, the term "*the closest clusters*" (according to the criterion of minimum Δ_{ij}) is approximately equivalent to "*the spatially closest clusters*" formulation (according to the criterion of minimum $\left\|\underline{\nu}_i - \underline{\nu}_j\right\|^2$). That is, in the beginning, the relative spatial localization of the clusters, given by each centroid position, has an important weight in establishing the optimal merging.

This is in fact the idea the LGS clustering uses to reduce complexity: *limit the searching neighborhood in the beginning, and gradually release this constraint as the clustering process advances.*

2.2 Proposed Algorithm

At each lth step of LGS algorithm, the corresponding clusters are pre-classified by a lattice whose generating matrix

$$\Lambda_l = \{\lambda_l(x,y)\}_{x,y=0,1,\ldots,K-1} \tag{2}$$

is given by

$$\lambda_l(x,y) = \begin{cases} 2^l & \text{if } x = y \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The diagonal matrix Λ_l generates a lattice that divides the K-dimensional space in hypercubes whose volume grows with each step of the algorithm. The particular choice of Λ_l was taken in order to:

- assure a fast cluster pre-classification, involving only shifts and comparisons;
- allow a successive decrease of pre-classification resolution that relaxes the constraint over the neighborhood where the search for optimal merging is performed.

Consequently, the algorithm is able to capture during its first steps the intimate arrangement of training vectors, by searching for the two closest clusters only in small hypercubes. Then, as the number of existing clusters decreases, the search is performed in larger hypercubes.

We assume now that a training sequence of $T_0 = T$ vectors must be partitioned into N clusters, and that the training vector components are scaled in the range $(0, 2^p - 1)$, where p is an integer and $p > 1$. Working on this training sequence, LGS clustering requires maximum p steps. The algorithm starts with a separate cluster for each training vector, initializes $l = 1$, and executes the lth step until the stopping criterion is achieved. The lth step of the algorithm is given by:

1. Find all hypercubes generated by Λ_l which contain at least two clusters.

2. For all selected hypercubes, search for the two closest clusters *located in the same hypercube* and merge them.

3. Do this search-and-merge operation until the number of clusters is reduced to a fraction $T_l = \xi^l T_0$ (where $0 < \xi < 1$), or there are no more hypercubes containing more than one cluster, or the number of clusters becomes $T_l = N$.

4. If $T_l = N$ then **Stop**, else let $l = l + 1$ and go to **1**.

Note that at any step of the algorithm, a specific cluster belongs always to only one hypercube. This is because an lth step hypercube is made up of 2^K hypercubes corresponding to the step $(l-1)$ (a direct consequence of the particular choice of the generating matrix Λ_l).

3 EXPERIMENTAL RESULTS

A practical implementation of LGS clustering requires a hierarchical data structure that contains preclassification information, regarding the clusters and the hypercubes they belong to.

LGS clustering has been tested for image vector quantization at a bit rate of 0.5 bits/pixel. Experiments have been performed on a PC-486/50 with simulation programs written in C language. We used 512×512 pixel monochrome still images with 8 bits/pixel ($p = 8$), and a vector dimension of $K = 4 \times 4 = 16$. The quality of reconstructed images has been evaluated by the Peak Signal to Noise Ratio ($PSNR$), defined as (dB)

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \tag{4}$$

where MSE denotes the Mean Squared Error between the original and the reconstructed image.

As an example, Figure 1 contains the original image *Lenna*, and Figure 2 presents the same image coded with a codebook derived via LGS algorithm ($PSNR = 31.80\ dB$). The training vector set was filled out with the $T = 16,384$ image vectors. Employing LGS with $\xi = 0.5$, the codebook of 256 codewords has been obtained in less than 100 seconds. For the same training sequence, the LBG algorithm initialized by choosing regularly spaced vectors, converged in 21 iterations, that is around $1,500$ seconds, yielding a codebook of similar quality ($PSNR = 31.60\ dB$).

4 CONCLUSIONS

A non-iterative algorithm for vector quantization clustering has been proposed in this paper. Using the hierarchical clustering concepts, the LGS algorithm efficiently finds the two closest clusters for merging by taking advantage of a cluster pre-classification operation. The pre-classification is obtained by inducing a growing lattice structure in the K-dimensional space, and limiting the searching neighbourhood to the clusters located in the same hypercubes.

Computer simulation results showed that the LGS clustering runs significantly faster than the classical LBG clustering, without sacrificing performance.

Figure 1: Original Image *Lenna*.
512×512 pixels, 8 bits/pixel.

Figure 2: Image *Lenna* Coded with LGS Codebook.
Bit Rate = 0.5 bits/pixel, $PSNR = 31.80$ dB.

References

[1] C. K. Chan and C. K. Ma, *A Fast Method of Designing Better Codebooks for Image Vector Quantization*, IEEE Trans. Commun., Vol. 42, No. 2/3/4, Feb./March/April 1994, pp. 237-242.

[2] D. Comaniciu, *An Efficient Clustering Algorithm for Vector Quantization*, Proc. 9th Scandinavian Conf. on Image Analysis, Uppsala, Sweden, June 1995, pp. 423-430.

[3] D. Comaniciu and R. Grisel, *Transform Vector Quantization with Training Set Synthesis*, Proc. 3rd Conf. on Digital Image Computing, Brisbane, Australia, Dec. 1995, pp. 139-144.

[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[5] W. H. Equitz, *A New Vector Quantization Clustering Algorithm*, IEEE Trans. ASSP, Vol. 37, No. 10, October 1989, pp. 1568-1575.

[6] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.

[7] C. M. Huang and R. W. Harris, *A Comparison of Several Vector Quantization Codebook Generation Approaches*, IEEE Trans. Image Processing, Vol. 2, No. 1, Jan. 1993, pp. 108-112.

[8] C.-H. Lee and L.-H. Chen, *A Fast Search Algorithm for Vector Quantization Using Mean Pyramids of Codewords*, IEEE Trans. Commun., Vol. 43, No. 2/3/4, Feb./March/April 1995, pp. 1697-1702.

[9] Y. Linde, A. Buzo, and R. M. Gary, *An Algorithm for Vector Quantizer Design*, IEEE Trans. Commun. Vol. COM-28, January 1980, pp. 84-95.

[10] V. J. Mathews, *Multiplication Free Vector Quantization using L1 Distortion Measure and Its Variants*, IEEE Trans. Image Processing, Vol. 1, No. 1, Jan. 1992, pp. 11-17.

[11] J. H. Ward, Jr., *Hierarchical Grouping to Optimize an Objective Function*, J. Amer. Statist. Ass., Vol. 58, 1963, pp. 236-244.

[12] X. Wu and L. Guan, *Acceleration of the LBG Algorithm*, IEEE Trans. Commun., Vol. 42, No. 2/3/4, Feb./March/April 1994, pp. 1518-1523.

[13] K. Zeger, A. Bist, and T. Linder, *Universal Source Coding with Codebook Transmission*, IEEE Trans. Commun., Vol. 42, No. 2/3/4, Feb./March/April 1994, pp. 336-346.