An Improved LBG Algorithm for Vector Quantization

Lecturer Cornel Balint,
„Politehnica" University of Timişoara

ABSTRACT. Vector quantization cover several fields such audio and video compression. This paper present a new method for the codebook generation, based of the classical LBG algorithm. A measure of utility for each code vector is used in order to improvement the codebook. This improvement tries to solve the problem of the local minima in a suboptimal codebook design due to an inadequate choice of the initial random codebook. Experiments with artificial data and with speech signals demonstrate improvements in terms of RMSE with no essential higher computational cost.

1 Introduction

Vector quantization VQ [GG92, Ger82] is an important and powerful technique for data compression. Speech and image signals compression, pattern recognition and computer vision are the usual applications of vector quantization.

Vector quantization and more generally clustering techniques use a set (or codebook) of reference vectors (or codeword) derived from a data set, named training set. Codeword are determined in order to minimize an objective function representing the quantization error. Using the codebook, each vector element of the input data is represented by one codeword.

An accepted classification scheme subdivides this clustering or vector quantization techniques in two main groups: hard (net) or soft (fuzzy). The difference between these two groups is the degree of membership of each vector to a cluster. In the case of hard techniques, each vector belongs to only one cluster, with a membership degree equal to unity. In the case of

fuzzy techniques, each vector can belong to several different clusters, with different degree of membership.

Another classification scheme introduces another two main group: K-means and competitive learning. K-means clustering techniques determine the codebook and respectively the codeword trying to minimize the average distortion over a suitable choice of the vector codeword. In the competitive learning techniques, inspired from neural network, the codeword are the result of a competition process between more candidate codewords.

The performance of the many vector quantization algorithms depends essentially of the choice of the initial codebook. A large number of quantization method vas developed in literature in order to solve this problem [GER92].

2 Vector quantization

A vector quantizer (VQ) of dimension k and size N consist in a mapping form an input vector (or point) in k-dimensional, Euclidean space, P^k, into a finite set X, containing N output points, called code vector or codeword:

$$Q: \mathrm{R}^k \to \mathrm{C}, \quad \mathrm{C} \in \mathrm{R}^k \tag{1}$$

$$\mathrm{C} = (\mathbf{y}_1, \mathbf{y}_2 .... \mathbf{y}_N) \text{ and } \mathbf{y}_i \in \mathrm{R}^k, \text{ for each } i \in \boldsymbol{I} = \{1,2,3,...N\}, \tag{2}$$

where I is a set of integer called index.

The set X represent the codebook (or simply the code) and contains N distinct elements, each of them vector in R^k.

Associated with VQ is a partition of R^k in N regions, named cells.

The i-th cell is defined as:

$$R_i = \left\{ x \in \mathrm{R}^k : Q(x) = \mathbf{y}_i \right\}, \quad i \in \boldsymbol{I} \tag{3}$$

and from cell definition follow that:

$$\mathrm{R}^k : \bigcup_i R_i = \mathrm{R}^k \text{ and } R_i \bigcap R_j = \varnothing \tag{4}$$

VQ encoding consists in mapping an input vector from R^k to the index set I, and the decoder maps the index I into the reproduction set X. The index of selected codeword is transmitted (or stored) as binary word and the VQ decoder generates the reproduction vector as a quantized approximation of the input vector.

As VQ performance measure a distortion measure *d* is defined as a nonnegative cost function $d(\mathbf{x}, \hat{\mathbf{x}})$ associated with any input vector **x** and

corresponding reproduction vector $\hat{\mathbf{x}}$. Given the cost function $d(\mathbf{x},\hat{\mathbf{x}})$, the performance of VQ can be appreciated by the average distortion:

$$\bar{d} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} d(X_i, \hat{X}_i) \qquad (5)$$

A distortion measure should be appropriate to the purpose of VQ, should be computable in real time in order to guide the actual encoding process and should be also subjective, so that large or small average distortion value should be correlate with bad or good subjective quality as perceived by the user of VQ system.

The most convenient and widely used measure of distortion is the squared Euclidian distance between the input and quantized vectors, defined as:

$$d(\mathbf{X},\hat{\mathbf{X}}) = \left\|\mathbf{X}-\hat{\mathbf{X}}\right\|^2 = \sum_{i=1}^{k} (X_i - \hat{X}_i)^2 \qquad (6)$$

Another particular distortion measure of practical interest is described in [GER92].

3 Nearest neighbor VQ

An important class of VQ that is of particular interest in a wide range of applications, called *nearest neighbor* (NN) (or *Voronoi*), has the feature that the partition of the input space is completely defined by the codebook and the distortion measure. In fact, the term *vector quantizer* is commonly assumed to be synonymous with *nearest neighbor vector quantizer*.

For a given distortion measure $d(\mathbf{x},\hat{\mathbf{x}})$, a *nearest neighbor* (NN) VQ is defined to be the VQ whose partition cells are given by:

$$R_i = \left\{ \mathbf{x} : d(\mathbf{x},\mathbf{y}_i) \le d(\mathbf{x},\mathbf{y}_j), \quad \forall j \in \boldsymbol{I} \right\} \qquad (7)$$

According to (7), in NN VQ each cell R_i consist of all point $\mathbf{x}$ which have less distortion relative to the reproduction vector $\mathbf{y}_i$ than with any other code vector.

The direct encoding algorithm for a NN VQ is given by the following:

Step 1: initialization: $d = d_0$, $j = 1$, $i = 1$,
Step 2: compute $D_j = d(x, y_j)$,
Step 3: if $D_j < d$, *set* $Dj \rightarrow d$ *and set* $j \rightarrow i$,
Step 4: if $j < N$ *then set* $j + 1 \rightarrow j$ *and go to step 2,*
Step 5: if $j = N$, *stop. Result is index i.*

The goal in design a optimal VQ is to find a codebook and a partition (or encoder rule) that will maximize the overall measure of performance. In [GG92] are given and demonstrated two necessary conditions for an optimal VQ:

a) nearest neighbor condition: for a given codebook X, the optimal partition cells satisfy the nearest neighbor condition (7) and:

$$\hat{\mathbf{x}} = Q(\mathbf{x}) \text{ only if } d(\mathbf{x}, \mathbf{y}_i) \le d(\mathbf{x}, \mathbf{y}_j), \forall j . \quad (8)$$

b) centroid condition: for a given partition R_i the optimal code vector satisfy:

$$\mathbf{y}_i = E\left[\mathbf{X} \middle| \mathbf{X} \in \mathrm{R}_i\right] \quad (9)$$

that is a generalization of the centroid:

$$\mathbf{y}_i = cent(\mathrm{R}_i) \quad (10)$$

4 LBG algorithm

An improvement of the simple nearest neighbor algorithm, proposed in [LBG80], consist of a finite sequence of iterations, in witch, at every step, a new quantizer, with a total distortion less or equal to the previous one, is produced.

The LBG algorithm suppose an initial codebook that is in the most case randomly generated. The algorithm consists in the following steps:

- *step 1: given the initial codebook, compute a partition of the input space according to the NN condition (7)*
- *step 2: given the partition computed in step 1, a new codebook is computed according centroid condition (9).*

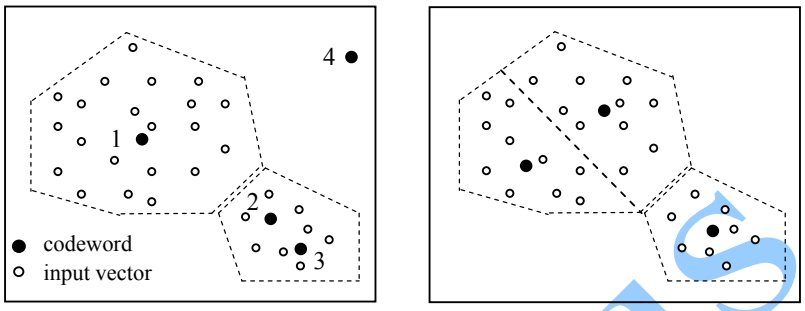
The algorithm stops when the quantizer distortion remains unchanged.

The LBG algorithm finally finds a local optimum quantizer. That quantizer depends essentially of the initial codebook and is often very far for an acceptable solution, a bad initialization could lead to the impossibility of finding a good quantizer [GK82].

An example is illustrated in fig. 1.

In fig. 1 a), two different situations that strongly limit the classical LBG algorithm are illustrated. The codeword number 4 will always generate an empty cell because all the input vectors are nearer to the other codewords and, according to LBG algorithm, it cannot move and never represent any input element. The configuration in fig. 1 show two clusters and three codewords: one codeword in the big cluster and two in the little one. The

input elements in the little cluster are well approximated by the two codewords whereas a lot of elements in the large cluster are badly approximated by the unique related codeword.



a) b)

Figure 1. Cells and codewords

5 An improvement of the LBG algorithm

In order to avoid the limitation of classical LBG algorithm, we introduce an improvement based on the previous improvements describes in [Fri97] and [PR01].

First, on compute the mean quantization distortion over all cells:

$$D_m = \frac{1}{N}\sum_{i=1}^{N} D_i \tag{11}$$

and define, for each cell, a utility measure U_i, as the value of distortion D_i of the i-th cell, normalized with respect to the mean value D_m:

$$U_i = \frac{D_i}{D_m} \tag{12}$$

For an optimum VQ, according to Gersho theorem [Ger79] each cell have the same contribution to the total distortion and consequently the utility index for all cell is equal to unity.

The equalization of cells utility is obtained by joining each cell with a low utility index with a cell adjacent to it, e trying to obtain a bigger cell with high utility. At the same time, each high utility cell will be split in two smaller cells, equivalent to moving one codeword from the low utility cell inside the high utility cell, as in fig. 1. b).

The split operation, illustrated in fig. 2, suppose that a cell is contained in a k – dimensional hyperbox defined by:

$$I = [x_{1m}, x_{1M}] \times [x_{2m}, x_{2M}] \times .... \times [x_{km}, x_{kM}] \quad (13)$$

where x_{im} and x_{iM} are the minimum and the maximum value by the i dimension of the cell. The two new codeword are placed on the principal diagonal of the hyperbox I at equal distance of the hyperbox center.

The improvement procedure selects each cell with a lower utility (lower that 1), in a sequential manner, and as improvement of previous algorithm we propose to try to minimize the overall distortion by using each cell with a utility higher that 1.

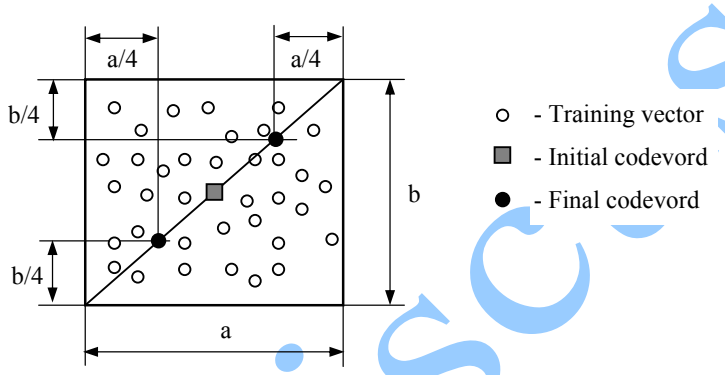


Figure 2. Splitting a codeword

For this, we compute the old distortion, before the codeword movement:

$$D_{old} = D(\{Y, C\}) \quad (14)$$

and the new distortion corresponding to the new partition Y':

$$D_{new} = D(\{Y', P(Y')\}) \quad (15)$$

for all the moving possibility.

The new situation with the minimum D_{new} is retained.

6 Simulations and results

The proposed improvement to the LBG algorithm was implemented in Matlab.

As test signal was used a artificial signal based on polynomial case (similar to the signal used in [PR01]) and sampled speech signal containing 4.000 – 50.000 samples, representing isolated phonemes and words [PA93, CRL92].

Execution time for the proposed algorithm compared with classical LBG was higher. To avoid the dependence of the reported execution time of the processor speed, we appreciate this time relative to LBG algorithm:

Codebook size	LBG		Proposed algorithm	
	Time [%]	Total RMSE	Time [%]	Total RMSE
128	100%	0,0458	146%	0,0401
256	205%	0,0302	293%	0,0289
512	410%	0,0142	600%	0,0140
1024	815%	0,0068	1275%	0,0061

The increase of computational time is not very high and must be appreciate considering that the codebook are constructed only once for a given VQ and then used for the quantization.

References

[CRL92] **D. Cohn, E. Riskin, R. Ladner** – *Theory and Practice of Vector Quantizers Trained on Small training Sets*, Tehnical Report TR 92-12-08, University of Washington, 1992

[Fri97] **B. Fritzke** – *The LBG-U Method for vector Quantization – an Improvement over LBG inspired from Neural Network*, Neural Processing Letters, vol. 5, no. 1, 1997

[Ger79] **A. Gersho** – *Asymptotically Optimal Bloc Quantization*, IEEE Trans. Inf. Theory, IT vol. 25, apr. 1979.

[Ger82] **A. Gersho** – *On the Structure on Vector Quantizers,* IEEE Trans. Inf. Theory, IT vol. 28, march 1982.

[GG92] **A. Gersho, R. M. Gray** – *Vector quantization and Signal Compression*, Kluwer Academic Publishers, London 1992

[GK82] **R. M. Gray, E. Karnin** – *Multiple Local Optima in Vector Quantizer*, IEEE Trans on Inform. Theory IT vol. 28, march 1982.

[LBG80] **Y. Linde, A. Buzo, R. M. Gray** – *An Algorithm for Vector Quantizer Design*, IEEE Transaction on Communications, vol. 28, 1980

[PA93] **K. Paliwal, B. Atal** – *Efficient Vector Quanization of LPC Parameters at 24 bit/frame*, IEEE Transaction on Speech and Audio Signal Processing vol. 1, no. 1, 1993

[PR01] **G. Patane, M. Russo** – *The enhanced LBG Algorithm,* Neural Networks, vol. 14 (no. 9), nov. 2001