1-1-1997

On the computational complexity of the LBG and PNN algorithms

Jamshid Shanbehzadeh
Tarbiat Moalem University, Islamic Republic of Ira

Philip Ogunbona
University of Wollongong, philipo@uow.edu.au

On the computational complexity of the LBG and PNN algorithms

Abstract

This correspondence compares the computational complexity of the pair-wise nearest neighbor (PNN) and Linde–Buzo–Gray (LBG) algorithms by deriving analytical expressions for their computational times. It is shown that for a practical codebook size and training vector sequence, the LBG algorithm is indeed more computationally efficient than the PNN algorithm.

Keywords

computational, algorithms, complexity, lbg, pnn

Disciplines

Physical Sciences and Mathematics

Publication Details

Shanbehzadeh, J. & Ogunbona, P. (1997). On the computational complexity of the LBG and PNN algorithms. IEEE Transactions on Image Processing, 6 (4), 614-616.

[9] G. Ramponi, "Edge extraction by a class of second-order nonlinear filters," *Electron. Lett.*, vol. 22, pp. 482–484, Apr. 24, 1986.
[10] G. L. Sicuranza, "Quadratic filters for signal processing," *Proc. IEEE*, vol. 80, pp. 1261–1285, Aug. 1992.
[11] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis.* Reading, MA: Addison-Wesley, 1991.
[12] J. D. Taft and N. K. Bose, "Quadratic linear filters for signal detection," *IEEE Trans. Signal Processing*, vol. 39, pp. 2557–2559, Nov. 1991.
[13] R. E. Williamson, H. F. Trotter, *Multivariable Mathematics.* Englewood Cliffs, NJ: Prentice-Hall, 1974.

On the Computational Complexity of the LBG and PNN Algorithms

J. Shanbehzadeh and P. O. Ogunbona

***Abstract*—This correspondence compares the computational complexity of the pair-wise nearest neighbor (PNN) and Linde–Buzo–Gray (LBG) algorithms by deriving analytical expressions for their computational times. It is shown that for a practical codebook size and training vector sequence, the LBG algorithm is indeed more computationally efficient than the PNN algorithm.**

I. Introduction

The most computationally complex part in the design of a vector quantization-based (VQ-based) [1], [2] coder/decoder system is the codebook generation. A popular scheme for VQ codebook generation is the Linde-Buzo–Gray (LBG) algorithm [3]. The LBG algorithm iteratively uses a given training sequence and an initial codebook to generate a locally optimum codebook. In an attempt to reduce the computational complexity of the LBG algorithm, Equitz [4] introduced the pair-wise nearest neighbor (PNN) clustering algorithm, and a suboptimum version (fast PNN) as a fast alternative scheme for codebook generation. He experimentally showed that the fast PNN algorithm is computationally more efficient than the LBG algorithm in terms of codebook generation. The comparison between PNN and LBG algorithms, however, was not considered.

This correspondence addresses the situation: We have compared PNN and LBG algorithms in terms of computational complexity. In Sections II and III, expressions for the computation times of LBG and PNN algorithms, respectively, are derived. In Section IV, the codebook generation times of these two schemes are compared. The final section contains our conclusion.

The symbols used in this correspondence are defined as follows.

m	Codebook size.
k	Codevector size.
t	Number of vectors in the training sequence.
n:	Number of iterations needed to meet stopping criterion.
η_{LBG}	Computation time for finding the distance between two vectors.
T_{com}	The unit time for comparing two distortion values.

Manuscript received September 9, 1995; revised June 5, 1996. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Amy R. Reibman.

The authors are with the Department of Electrical and Computer Engineering, University of Wollongong, NSW 2522 Australia (e-mail: pogna@elec.uow.edu.au).

Publisher Item Identifier S 1057-7149(97)02458-5.

II. LBG Algorithm: Computation Times

The LBG algorithm is an iterative procedure where, at each iteration, each vector in the training sequence is compared with all the codevectors in the current version of the codebook and assigned to the same cluster as the most similar codevector. Normally, the criterion of similarity is the mean square error (MSE) or weighted MSE. After each iteration the sequence of codevectors is updated by using the moment center of each cluster. The stopping criterion is satisfied when the ratio of the overall distortion improvement to the distortion in the previous iteration is less than a predefined value.

It is easy to show that the total computation time for the LBG algorithm is

$$T_{\mathrm{LBG}} = A_{\mathrm{LBG}} + B_{\mathrm{LBG}} \tag{1}$$

where

$$A_{\mathrm{LBG}} = nmt\eta_{\mathrm{LBG}}$$
$$B_{\mathrm{LBG}} = n(m-1)tT_{\mathrm{com}}.$$

A_{LBG} is the distortion computation time and B_{LBG} is the time for comparing distortions. A proof of (1) is as follows: There are t vectors in the training sequence, each of which is to be compared with all the codevectors. The number of iterations is n, and at each iteration we need to find the minimum among the m distortion values computed; this requires $(m-1)$ comparisons. Thus, $A_{\mathrm{LBG}} = nmt\eta_{\mathrm{LBG}}$ and $B_{\mathrm{LBG}} = n(m-1)tT_{\mathrm{com}}$. If the MSE is considered as the distance measure, then the required operations for calculating η_{LBG} are $(2k-1)$ additions and k multiplications. If we further assume that the CPU time required for multiplication is α times more than that for addition $(t\mathrm{add})$, we can write

$$\eta_{\mathrm{LBG}} = ((2+\alpha)k-1)t_{\mathrm{add}}. \tag{2}$$

III. PNN Algorithm: Computation Time

In the first step of the PNN algorithm, each vector of the training sequence is considered as a cluster and the distortion between every pair of vectors is computed. The pair with the least distortion are merged. At each of the subsequent steps, two clusters that result in the least possible average distortion are merged. In order to find out the merging that results in the least distortion, all the distortion information arising from merging any pair of clusters are required. A formula that gives the distortion arising from merging two clusters (i and j) based on MSE [4] is

$$\text{Distortion} = n_i S_i^2 + n_j S_j^2 + \frac{n_i n_j}{n_i + n_j}|\overline{x}_i - \overline{x}_j|. \tag{3}$$

n_i and n_j are, respectively, the numbers of vectors in clusters i and j. S_i^2 and S_j^2 are the average distortion (MSE) computed over the vectors in clusters i and j, and their respective cluster centers ($\overline{x}_i$ or $\overline{x}_j$).

The computational burden incurred in the first step of PNN algorithm has two parts: i) computation of the distortion between each pair of vectors and ii) the cost of comparisons. With t number of training vectors, the computational cost of distortion calculation is proportional to $\binom{t}{2} = (t^2-t)/2$. The computational cost of comparing the $t^2 - t/2$ calculated MSE is $[((t-1)^2 - (t-1))/2]T_{\mathrm{com}}$. After merging two clusters we are left with $t-1$ clusters. The distortion between the newly generated cluster and the rest is

calculated. This results in $t-2$ distortion calculations and the time for comparing all the distortions is $[((t-2)^2-(t-2))/2]T_{\text{comm}}$. For a target codebook size, m, this process continues until m clusters are obtained. The required time for the first step of PNN is easily written as

$$t_1 = \frac{t^2-t}{2}\eta_{\text{LBG}} + \frac{(t-1)^2-(t-1)}{2}T_{\text{comm.}} \tag{4}$$

In the ith step, the computational time is

$$t_i = (t-i)\eta_{\text{PNN}} + \tfrac{1}{2}[(t-i-1)^2-(t-i-1)]T_{\text{COM}} \tag{5}$$

where η_{PNN} is the time for computing the distortion measure in (3); this distortion measure is calculated for each cluster, and the newly generated cluster from the previous step. This time can be calculated by considering (3): If the information about S_i^2 and S_j^2 are available, the required operations are four multiplications, one division, two additions, and the rest is the same as η_{LBG} in the LBG algorithm. If we assume that the CPU time for division is β times greater than that for addition, then η_{PNN} can be written as

$$\eta_{\text{PNN}} = (4\alpha+\beta+(2+\alpha)k+1)t_{\text{add.}} \tag{6}$$

The overall time for PNN is then

$$T_{\text{PNN}} = A_{\text{PNN}} + B_{\text{PNN}} \tag{7}$$

where

$$A_{\text{PNN}} = \sum_{i=1}^{t-m}(t-i)\eta_{\text{PNN}} + \tfrac{1}{2}(t^2-t)\eta_{\text{LBG}}$$

$$B_{\text{PNN}} = \tfrac{1}{2}\sum_{i=1}^{t-m}((t-i)^2-(t-i))T_{\text{com.}}$$

After some manipulation of (7), we have

$$A_{\text{PNN}} = \tfrac{1}{2}[t^2-t-m^2+m]\eta_{\text{PNN}} + \tfrac{1}{2}(t^2-t)\eta_{\text{lBG}} \tag{7.1}$$

$$B_{\text{PNN}} = [t^3+(-5m+\tfrac{5}{2})t^2+(5m^2-5m+\tfrac{3}{2})t - (2m^2+\tfrac{5}{2}m-\tfrac{3}{2})m]T_{\text{com.}} \tag{7.2}$$

IV. Comparison of Computational Times and Objective Performance

In this section, we derive the constraint under which the LBG algorithm becomes computationally more efficient than PNN. This constraint is then tested in practical situations.

It is clear that

$$A_{\text{LBG}} \le A_{\text{PNN}} \quad \text{and} \quad B_{\text{LBG}} \le B_{\text{PNN}} \Rightarrow T_{\text{LBG}} \le T_{\text{PNN.}} \tag{8}$$

Furthermore, $A_{\text{LBG}} \le A_{\text{PNN}}$ is equivalent to

$$nmt\eta_{\text{LBG}} \le \tfrac{1}{2}[t^2-t-m^2+m]\eta_{\text{PNN}} + \tfrac{1}{2}(t^2-t)\eta_{\text{LBG.}} \tag{9}$$

If both sides of (9) are divided by $mt\eta_{\text{LBG}}$, the result, after some straightforward manipulations, is

$$n \le \frac{1}{2}\left(\left(\frac{t}{m}-\frac{1}{m}\right)\left(1+\frac{\eta_{\text{PNN}}}{\eta_{\text{LBG}}}\right)+\left(\frac{1}{t}-\frac{m}{t}\right)\right). \tag{10}$$

Now, in practice $t \gg 1$ and $t \gg m$, hence (10), can be written as

$$n \le \frac{1}{2}\left(\frac{t}{m}-\frac{1}{m}\right)\left(1+\frac{\eta_{\text{PNN}}}{\eta_{\text{LBG}}}\right). \tag{11}$$

If the expressions for η_{PNN} and η_{LBG} from (2) and (6) are substituted into (11), we have

$$n \le \frac{1}{2}\left(\frac{t}{m}-\frac{1}{m}\right)\left(1+\frac{4\alpha+\beta+(2+\alpha)k+1}{(2+\alpha)k-1}\right). \tag{12}$$

After some manipulations, the inequality in (12) can be written as

$$n \le \left(\frac{t}{m}-\frac{1}{m}\right)\left(1+\frac{2\alpha+\frac{\beta}{2}+2}{\left(1+\frac{\alpha}{2}\right)k-\frac{1}{2}}\right). \tag{13}$$

We now use the fact that α, β, and k are all greater than unity, and $t \gg m$ to obtain the required constraint for $A_{\text{LBG}} \le A_{\text{PNN}}$ in a simple form

$$n \le \frac{t}{m.} \tag{14}$$

We now turn our attention to the constraint required for $B_{\text{LBG}} \le B_{\text{PNN}}$. $B_{\text{LBG}} \le B_{\text{PNN}}$ is equivalent to (15), shown at the bottom of the page. Again, we use the fact that in practice $t \gg 1$ and $t \gg m$ to simplify (15) into

$$n < \frac{t^2}{m-1.} \tag{16}$$

Next, we show that if (14) is true, then (16) is also true. With $t > 1$

$$n \le \frac{t}{m} \Rightarrow n \le \frac{t^2}{m}. \tag{17}$$

It then follows that

$$\frac{t^2}{m} \le \frac{t^2}{m-1} \Rightarrow n \le \frac{t^2}{m-1}. \tag{18}$$

Hence, it is sufficient to show that (14) is true to prove that the LBG is computationally more efficient than PNN. In order to evaluate the codebook generation times of LBG and PNN, the results quoted by Equitz [4] are used. These results are shown in Table I. The size of all the test images are 512 × 512 pixels. The codebook size is 256 and the vector size is 16. The first column of Table I is the image under test, the second column is the number of LBG algorithm iterations when the initial codebook is selected randomly, the third column is t/m, and the final column is the ratio of LBG codebook generation time over fast PNN obtained by Equitz [4]. It can be easily seen that for all cases, LBG requires less codebook generation time. However, in case of codebook generation time, fast PNN is much superior than LBG.

The main goal of each clustering algorithm is to achieve the best possible partitioning of the data based on a classification measurement. Here, the performance achieved by the LBG and PNN algorithms for some sample images are shown in Table II. The tests have been performed on 128 × 128 images and an image-based codebook of size 64. The images are part of the some test images from Table I, and the original images and the tested parts are shown in Fig. 1. The reasons for using part of the images are twofold; the computational time of PNN algorithm in practice and the memory

$$n < \frac{\left[t^3+\left(-5m+\frac{5}{2}\right)t^2+\left(5m^2-5m+\frac{3}{2}\right)t-\left(2m^2+\frac{5}{2}m-\frac{3}{2}\right)m\right]}{(m-1)t} \tag{15}$$

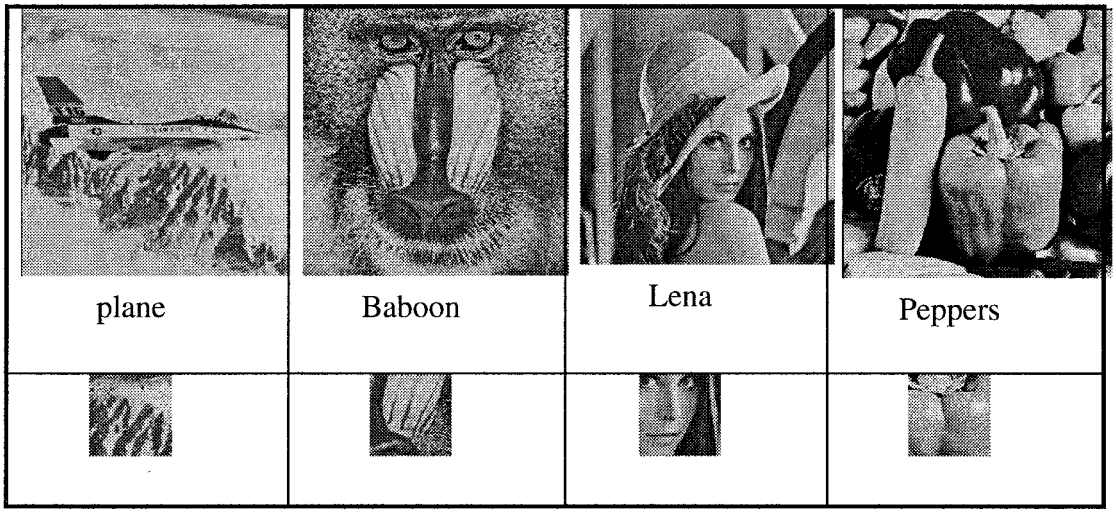

Fig. 1. Test images for objective quality comparison of LBG and PNN algorithms.

TABLE I
COMPARISON AMONG THE CODEBOOK GENERATION TIMES OF LBG, PNN AND FAST PNN ALGORITHMS

Image	LBG iterations with random initialisation	$\frac{t}{m}$	Ratio of LBG Execution time over fast PNN
Baboon	17	64	27.76
Lake	25	64	23.26
Airport	27	64	43.13
Lena	25	64	18.93
Peppers	31	64	23.63
Plane	33	64	23.94

required to execute PNN are very high. The memory required for a typical 512×512 image and a vector size of 16 in situation, where 4 bytes are used for measuring the distortion between two clusters will be more than 1 Gbytes. If we use the hard disk of computer for memory, it will heavily affect the real execution time of the PNN algorithm. PSNR, based on the following formula, has been used to measure the objective performance of the two algorithms.

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{mse}} \right). \tag{19}$$

It can be seen that in all cases, the LBG algorithm shows better performance than the PNN algorithm, and in fact LBG has a better performance than the suboptimum version of PNN (fast PNN) as well.

V. CONCLUSION

In this correspondence, the computational times and the objective quality performance of two codebook generation schemes, LBG and PNN, have been compared. The constraint under which the LBG algorithm is computationally more efficient than the PNN algorithm was theoretically obtained, and it is shown that, in practice, the codebook generation time of LBG is indeed less than that of PNN. The performance of the LBG algorithm for all the test images was better than the PNN algorithm. We hasten to point out that the widely held opinion about the superiority of PNN over LBG is only true for the suboptimum version of PNN, fast PNN.

TABLE II
PSNR OF LBG AND PNN ALGORITHMS

Image	PSNR LBG (dB)	PSNR PNN (dB)
Peppers	28.79	28.16
Baboon	25.40	24.20
Lena	30.54	29.35
Plane	28.41	26.35

REFERENCES

[1] R. M. Gray, "Vector quantization," *IEEE Acoust., Speech, Signal Processing Mag.*, pp. 4–29, Apr. 1984.
[2] J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, pp. 1551–1588, Nov. 1985.
[3] Y. Linde, A. Buzo, R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.
[4] W. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1568–1575, Oct. 1989.