Affine LBG Algorithm for Codebook Training of Univariate Linear Approximation

Tiannan Dong, Jianji Wang*, Meng Yang, Kai Yi, and Nanning Zheng

National Engineering Laboratory for Visual Information Processing and Applications
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University, Xi'an, Shaanxi 710049, P. R. China
Email: wangjianji@mail.xjtu.edu.cn

***Abstract*—LBG algorithm is a simple and effective method to train codebook for vector quantization. After LBG was proposed, several interesting algorithms were published to improve the effectiveness and efficiency of LBG. Unlike vector quantization, univariate linear approximation is another important data compression method, which approximates a target vector by a linear transformation of a selected codeword from codebook. Many applications also use LBG or K-means algorithm to train the codebook for univariate linear approximation. In this paper, we proposed an improved LBG algorithm called the affine LBG algorithm to train the codebook for univariate linear approximation. The experimental results show that the affine LBG algorithm can derive a more effective codebook than LBG algorithm for univariate linear approximation. Moreover, the A-LBG algorithm is more efficient than LBG algorithm.**

I. Introduction

LBG algorithm was proposed by Linde, Buzo, and Gray in 1980 [1], which is originally used to derive the codebook of vector quantization (VQ). As a classical data compression technology, vector quantization has been widely applied in various fields such as image processing, pattern recognition and audio processing [2].

In VQ, some groups of scalar data are put together to be a target vector, then the target vector is quantized by a codeword in a codebook. Suppose we have a codebook $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$, then we need to find a codeword in the codebook $\mathbf{V}$ to approximate the target vector $\mathbf{y}$. If the distance function used in VQ is $d(\cdot, \cdot)$, then the final selected vector $\mathbf{v}_i$ for the target vector $\mathbf{y}$ should satisfy the equation $d(\mathbf{v}_i, \mathbf{y}) \leq d(\mathbf{v}_j, \mathbf{y})$, $\mathbf{v}_i \in \mathbf{V}$, $j = 1, 2, \cdots, m$. Lastly, the target vector $\mathbf{y}$ is directly approximated by $\mathbf{v}_i$. Generally, the l-2 distance is used as the distance function.

The codebook $\mathbf{V}$ is highly associated with the target vector. $\mathbf{V}$ is usually trained from a lot of samples like $\mathbf{y}$.

The training of the codebook $\mathbf{V}$ is a very important work for VQ because the effectiveness of VQ in fact all depends on the codebook. The LBG algorithm is a simple and effective clustering method to train the codebook for VQ. The main thinking of LBG is very similar as the K-means algorithm, which has been deemed to be one of the top ten data mining algorithms.

After the LBG algorithm was proposed, researcher found several shortages of LBG. For example, LBG algorithm is easily trapped into a local minimum. Moreover, the number of clustering is hard to judge by LBG algorithm so we have to assign a value for it.

Many works have been conducted to improve LBG algorithm, and some interesting applications are also proposed. Firstly, many fast LBG algorithms were proposed to speed up the LBG process. Based on the statistical energy distribution of transformed training vectors, Pan *et al* proposed a fast LBG algorithm [3]. Lin and Tai also proposed a fast method based on correlation among the pixels to reduce more than 90% of the time [4]. Some strategies were proposed to improve the effectiveness of LBG. For example, Mirzaei *et al* offered an effective codebook initialization method [5] and Deussen *et al* provided a weighted LBG stippling [6]. Recent years, LBG has been applied in various field such as sign language recognition [7], steganography [8], and classification [9].

In VQ, a target vector is directly approximated by one codeword in the codebook. Although VQ is a simple and efficient method, its effectiveness is usually unsatisfied. In many applications, such as fractal image compression [10], a target vector is approximated by the linear transformation of a codeword in the codebook. For a codebook $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$, the target vector $\mathbf{y}$ is lastly approximated by the selected vector $\mathbf{v}_i$ as

$$\mathbf{y} = \lambda \mathbf{v}_i + o\mathbf{1} \tag{1}$$

where λ is the scalar multiplier, o is the offset and $\mathbf{1}$ is the vector with all ones. The value of λ and o can be computed by the least square method. Unlike VQ, here we call the question to use a linear transformation of a codeword to approximate the target vector as the "Univariate Linear Approximation (ULA)". It is clearly that the effectiveness of ULA is better than VQ.

Some applications also use LBG algorithm or K-means algorithm to train the codebook of ULA [11]. However, ULA differs from VQ.

In this paper, we propose an improved LBG algorithm which we call as the Affine LBG Algorithm (A-LBG) to construct

This work was supported in part by the National Key Research and Development Program of China under grant 2016YFB1000903, the key project of Trico-Robot plan of NSFC under grant No. 91748208, and the National Natural Science Foundation of China under Grant 91648208.

the codebook of ULA. The experimental results show that the codebook constructed by the proposed A-LBG algorithm is always more effective than the codebook constructed by LBG algorithm for an arbitrary initial codebook in ULA.

II. The Affine LBG Algorithm

Suppose we have a codebook $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$, the standard deviation of the elements in $\mathbf{v}_i$ is σ_i, and the mean of the elements in $\mathbf{v}_i$ is μ_i, $i = 1, 2, \cdots, m$. For an arbitrary target vector $\mathbf{y}$ to be encoded in univariate linear approximation (ULA), suppose the standard deviation and the mean of the elements in $\mathbf{y}$ are σ and μ, respectively. Generally, we have $\sigma_i \neq \sigma$ and $\mu_i \neq \mu$. Therefore, $\mathbf{y}$ cannot be directly encoded by a codeword. A scalar multiplier and an offset also need to be considered. Lastly, we encode $\mathbf{y}$ as in Eq. (1).

As mentioned above, the codebook $\mathbf{V}$ plays a key role in VQ as well as ULA. Hence, the training of the codebook is very important. LBG is a simple and effective algorithm to train the codebook of VQ. Here we consider how to train the codebook for ULA.

In LBG algorithm, if the number of clusters is k and the number of samples is n, then the steps of LBG is as following:

(1) k initial clustering centers need to be constructed firstly.

(2) For each sample, an clustering center needs to be chosen by minimizing the distance between this sample and this clustering center, and then this sample is classified into the cluster corresponding with the selected clustering center. All n samples are classified according to this method.

(3) For each cluster, the mean of the samples in this cluster are computed as the new clustering center of this cluster.

(4) Repeat steps (2) and (3) until a stopping condition is satisfied.

The above mentioned stopping condition may be an iterated number, a variation of the clustering centers and so on.

According to step (2), for a sample, an clustering center needs to be chosen by minimizing the distance between this sample and this clustering center in LBG algorithm for VQ. However, ULA is different with VQ because a target vector is approximated directly by a codeword in VQ, but a target vector is approximated by a linear transformation of the selected codeword in ULA. Therefore, in step (2), for a sample, an clustering center needs to be chosen by minimizing the distance between this sample and a linear transformation of this clustering center for ULA.

If the variance of a vector is zero, then the vector can be directly losslessly coded by the vector 1 without any codewords. Moreover, if the variance of a codeword in a codebook is zero, the codeword is useless because it is correlation with $\mathbf{1}$. Thus, here we only consider these samples whose variances are not zero.

Then a question arises: how to minimize the l-2 distance between a sample and a linear transformation of a clustering center from all the clustering centers? In fact, according to the references [12], we have known that the minimization of the l-2 distance between a sample and a linear transformation of a clustering center from all the clustering centers is equivalent to maximization of the absolute value of Pearson's inner-product correlation coefficient between this sample and a clustering center from all the clustering centers. According to this conclusion, the step (2) of the proposed algorithm to train the codebook of ULA is changed as follows:

For each sample, an clustering center needs to be chosen by maximizing the absolute value of Pearson's correlation coefficient between this sample and this clustering center, and then this sample is classified into the cluster corresponding with the selected clustering center. All n samples are classified according to this method.

Similarly, in step (3), the mean of the samples in each cluster are computed as the new clustering center of this cluster in LGB algorithm for VQ. In this case, all samples in the same cluster are added together, and then the addition is divided by the number of samples in this cluster. A default operation here is that each sample in this cluster is directly corresponded with the clustering center, which is just the approximation method in VQ. In ULA, each clustering center should be corresponded with a linear transformation of each sample in this cluster. If the current clustering center is $\mathbf{v}$, a sample in this cluster is $\mathbf{s}$, and $\lambda\mathbf{s} + o\mathbf{1}$ is the linear transformation which transform $\mathbf{s}$ to approximate the clustering center $\mathbf{v}$. Then we need to minimize the mean squared error (MSE) between $\mathbf{v}$ and $\lambda\mathbf{s} + o\mathbf{1}$:

$$\min \mathrm{MSE}(\mathbf{v}, \lambda\mathbf{s} + o\mathbf{1}) = \min \frac{1}{d} \|\mathbf{v} - (\lambda\mathbf{s} + o\mathbf{1})\|_2^2 \qquad (2)$$

where d is the dimension of these vectors and $\|\cdot\|_2$ is the 2-norm.

Suppose the means of $\mathbf{v}$ and $\mathbf{s}$ are $\mu_{\mathbf{v}}$ and $\mu_{\mathbf{s}}$, respectively, the standard deviations of $\mathbf{v}$ and $\mathbf{s}$ are $\sigma_{\mathbf{v}}$ and $\sigma_{\mathbf{s}}$, respectively, and the covariance between $\mathbf{v}$ and $\mathbf{s}$ is $\sigma_{\mathbf{vs}}$, then we have

$$\begin{aligned}
&\mathrm{MSE}(\mathbf{v}, \lambda\mathbf{s} + o\mathbf{1}) = \tfrac{1}{d} \|\mathbf{v} - (\lambda\mathbf{s} + o\mathbf{1})\|_2^2 \\
&= \tfrac{1}{d}\langle \mathbf{v} - \lambda\mathbf{s} - o\mathbf{1}, \mathbf{v} - \lambda\mathbf{s} - o\mathbf{1}\rangle \\
&= \tfrac{1}{d}\langle \mathbf{v}, \mathbf{v}\rangle + \tfrac{\lambda^2}{d}\langle \mathbf{s}, \mathbf{s}\rangle + o^2 - 2\lambda\langle \mathbf{v}, \mathbf{s}\rangle - \tfrac{2o}{d}\langle \mathbf{v}, \mathbf{1}\rangle + \tfrac{2\lambda o}{d}\langle \mathbf{s}, \mathbf{1}\rangle \\
&= \sigma_{\mathbf{v}}^2 - 2\lambda\sigma_{\mathbf{vs}} + \lambda^2\sigma_{\mathbf{s}}^2 + \mu_{\mathbf{v}}^2 - 2\lambda\mu_{\mathbf{v}}\mu_{\mathbf{s}} \\
&\quad + \lambda^2\mu_{\mathbf{s}}^2 + o^2 - 2o\mu_{\mathbf{v}} + 2\lambda o\mu_{\mathbf{s}}.
\end{aligned} \qquad (3)$$

To minimize MSE between $\mathbf{v}$ and $\lambda\mathbf{s} + o\mathbf{1}$,

$$\frac{\partial \mathrm{MSE}(\mathbf{v}, \lambda\mathbf{s} + o\mathbf{1})}{\partial o} = 0 \Rightarrow o = \mu_{\mathbf{v}} - \lambda\mu_{\mathbf{s}}. \qquad (4)$$

Substitute Eq. (4) to Eq. (3), we have

$$\mathrm{MSE}(\mathbf{v}, \lambda\mathbf{s} + o\mathbf{1}) = \sigma_{\mathbf{v}}^2 - 2\lambda\sigma_{\mathbf{vs}} + \lambda^2\sigma_{\mathbf{s}}^2. \qquad (5)$$

Then

$$\frac{\partial \mathrm{MSE}(\mathbf{v}, \lambda\mathbf{s} + o\mathbf{1})}{\partial \lambda} = 0 \Rightarrow \lambda = \frac{\sigma_{\mathbf{vs}}}{\sigma_{\mathbf{s}}^2}. \qquad (6)$$

For a sample in a cluster, Eqs. (4) and (6) offer the method to compute the parameters. Then in step (3), all samples are transformed near the clustering center by linear transformations. In LBG algorithm, we take the mean of these samples as the new clustering center of this cluster. In the proposed algorithm, the absolute values of Pearson's correlation coefficient is used to judge which cluster a sample belongs to. Because the mean

of some vectors and the addition of these vectors are totally correlation, here we can only use the addition of the linear transformations of samples to substitute the mean of them.

Finally, if the number of clusters is k and the number of samples is n, the steps of the proposed algorithm to train codebook for ULA is as following:

(1) k initial clustering centers need to be constructed firstly.

(2) For each sample, an clustering center needs to be chosen by maximizing the absolute value of Pearson's correlation coefficient between this sample and this clustering center, and then this sample is classified into the cluster corresponding with the selected clustering center. All n samples are classified according to this method.

(3) For each cluster, the addition of the linear transformations of the samples in this cluster are computed as the new clustering center of this cluster. The linear parameters of the linear transformations can be computed by Eqs. (4) and (6).

(4) Repeat steps (2) and (3) until a stopping condition is satisfied.

Because the proposed algorithm is an improved algorithm of LBG for ULA, and the affine transformations are used in steps (2) and (3) compared with LBG algorithm, we call the proposed algorithm as the affine LBG algorithm (A-LBG).

If some simple preprocessing can be made to the original samples, the proposed A-LBG algorithm can be further simplified. According to the above analysis, we only consider the samples with non-zero-variances. For a vector $\mathbf{v}$, if its mean and standard derivation are $\mu_{\mathbf{v}}$ and $\sigma_{\mathbf{v}}$, respectively, we denote $\mathbf{v}'$ the standard vector of $\mathbf{v}$:

$$\mathbf{v}' = \frac{\mathbf{v} - \mu_{\mathbf{v}}\mathbf{1}}{\sigma_{\mathbf{v}}}. \tag{7}$$

Obviously, the mean of $\mathbf{v}'$ is zero and the length of $\mathbf{v}'$ is 1. If all the samples and clustering centers are standard vectors, then according to Eqs. (4) and (6) we have

$$\begin{aligned} o &= 0 \\ \lambda &= r_{\mathbf{vs}} \end{aligned} \tag{8}$$

where $r_{\mathbf{vs}}$ is Pearson's correlation coefficient between $\mathbf{v}$ and $\mathbf{s}$. When $\mathbf{v}$ and $\mathbf{s}$ are both standard vectors, Pearson's correlation coefficient is just the inner product of them.

Then the steps of the simplified A-LBG algorithm is as following:

(1) Transform all the samples to be the standard vectors of these samples by Eq. (7).

(2) Construct k initial clustering centers and transform them to be standard vectors.

(3) For each sample, compute the absolute values of the inner products between this sample and all the clustering centers, then classify the sample into the cluster whose absolute value of inner product with the sample is maximum. All samples are classified according to this method.

(4) For a cluster, each sample in the cluster is transformed with multiplying by the inner product between the sample and the clustering center of this cluster. Then the addition of these transformed samples is computed, and we take the standard vector of the addition of these transformed samples as the new clustering center of this cluster.

(5) Repeat steps (3) and (4) until a stopping condition is satisfied.

Lastly, for a target vector $\mathbf{y}$ to be encoded in ULA, if the codebook trained by the proposed A-LBG algorithm is $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$ in which all the codewords are standard vectors, we compute the absolute values of the inner products between $\mathbf{y}$ and all the codewords in $\mathbf{V}$, then the codeword $\mathbf{v}_i$ corresponding the maximum absolute value of the inner product is the selected codeword for the target vector $\mathbf{y}$. Finally, the linear transform of $\mathbf{v}_i$ to approximate the target vector $\mathbf{y}$ is as following according to Eqs. (4) and (6):

$$\lambda\mathbf{v}_i + \mu\mathbf{1} \tag{9}$$

where λ is the inner product between $\mathbf{y}$ and $\mathbf{v}_i$, and μ is the mean of $\mathbf{y}$.

III. Experimental Analysis

Here we use both LGB algorithm and the proposed A-LGB algorithm to derive codebooks for ULA-based image encoding. The 512×512 image Lena is used to test the methods, and the image is divided into 4096 blocks with size 8×8, which are taken as the samples to train the codebook. When we train the codebook use LGB or A-LGB algorithm, all the samples are standardized according to Eq. (7). That is to say, when we use LGB or A-LGB algorithm to train the codebook, the samples are all zero-mean and the length of each sample is always 1.

The PSNR value of the reconstructed images against the iterated number of different algorithms is shown in Fig. 1. From Fig. 1 we can see that the reconstructed image quality with the codebook trained by A-LBG algorithm is better than the reconstructed image quality with the codebook trained by LBG algorithm. Here each codebook includes 256 codewords.

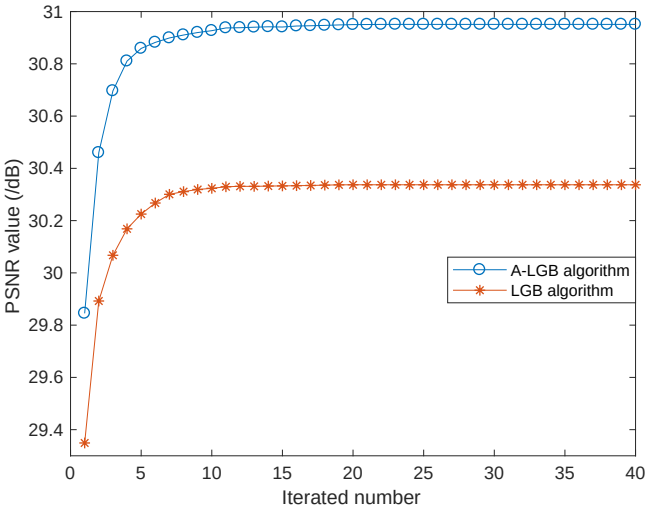

Fig. 1. The PSNR value of the reconstructed images of 512×512 Lena against the iterated number with different codebooks trained by LBG algorithm and A-LBG algorithm. 256 codewords are included in each codebook.

The reconstructed image with codebook trained by LBG algorithm

The reconstructed image with codebook trained by A-LBG algorithm

Fig. 2. The reconstructed images of 512×512 Lena with different codebooks trained by LBG algorithm and A-LBG algorithm. The standard vectors of the 4096 blocks with size 8×8 divided from image Lena are used as the samples to train the codebook in both LBG and A-LBG algorithms.

The reconstructed images of Lena with codebook trained in different algorithms when the iterated number is 50 are shown in Fig. 2. Here 256 codewords are included in the codebook. It can be seen from Fig. 2 that the texture of the reconstructed image encoded by the codebook trained in A-LBG algorithm is smoother, and the edge of the objects in the reconstructed image encoded by the A-LBG codebook is more natural. In fact, the PSNR value of the reconstructed image by the A-LBG codebook is 30.9509 dB, and it is 30.3372 dB of the reconstructed image by the LBG codebook.

Further experiments show that the cases like Figs. 1 and 2 are universal phenomenon. Suppose an image is divided into many vectors with the same size which are taken as the samples. When the number of codewords, the samples used to train the codebook, and the initial clustering centers are all kept the same and the number of codewords is less than the number of the samples with non-zero variance, the reconstructed image quality with the A-LBG codebook is always better than the reconstructed image quality with the LBG codebook. This experimental result shows the effectiveness of the proposed A-LBG algorithm for ULA questions.

In addition, A-LBG algorithm is more efficient than LBG algorithm. The main time complexity of LBG and A-LBG comes from the process of classification. In each iterated process of LBG or A-LBG algorithm, all samples needs to be compared with all the clustering centers. The number of comparisons is $n \times k$ in which n is the number of samples and k is the number of clusters. Suppose the dimension of the vectors is d. In LBG algorithm, when a sample is compared with a clustering center, we need to compute the l-2 distance between them, which needs $2d-1$ additions and d multiplications. In the proposed A-LBG algorithm, when a sample is compared with a clustering center, we only need to compute the inner product between them, which needs $d-1$ additions and d multiplications. Hence, in overall, in each iterated process the additions in LBG algorithm is $n \times k \times d$ more than the additions in A-LBG algorithm. In our experiment, the time to derive a codebook with 256 codewords from 4096 samples is about 4.7 s by the proposed A-LBG algorithm in matlab, but it is about 7.6 s by LBG algorithm.

IV. Discussion

The proposed A-LBG algorithm only add affine transformation in the steps (2) and (3) of LBG algorithm. Hence, it has the same shortages of LBG. However, the proposed A-LBG algorithm can be combined with almost all the strategies used in LBG algorithm to overcome these shortages.

V. Conclusion

LBG algorithm was proposed to derive the codebook for VQ. In this paper, we propose an improved LBG algorithm by adding some affine transformations in the steps of LBG algorithm. We call the proposed improved LBG algorithm as the affine LBG algorithm (A-LBG). The experimental results show that the proposed A-LBG algorithm is more effective and efficient for codebook training in univariate linear approximation (ULA) problem than LBG algorithm.

References

[1] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1), 1980, pp. 84-95.

[2] A. Gersho and R.M. Gray. (2012). Vector quantization and signal compression (Vol. 159). Springer Science & Business Media.

[3] Z.B. Pan, G.H. Yu, and y. Li. Improved fast LBG training algorithm in Hadamard domain. *Electronics letters*, 47(8), 2011, pp.488-489.

[4] Y.C. Lin, and S.C. Tai. A fast Linde-Buzo-Gray algorithm in image vector quantization. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(3), 1998, pp.432-435.

[5] B. Mirzaei, H. Nezamabadi-pour, and D. Abbasi-moghadam. An effective codebook initialization technique for LBG algorithm using subtractive clustering. *In Proc. 2014 IEEE Iranian Conference on Intelligent Systems (ICIS2014)*, 2014, pp. 1-5.

[6] O. Deussen, M. Spicker, Q. Zheng. Weighted linde-buzo-gray stippling. *ACM Transactions on Graphics*, 36(6), 2017, 233.

[7] K.S. Raut, S. Mali, S.D. Thepade, and S.P. Sanas. Recognition of American sign language using LBG vector quantization. *In Proc. 2014 IEEE International Conference on Computer Communication and Informatics (ICCCI2014)*, 2014, January, pp. 1-5.

[8] S.D. Thepade and S.S. Thube. Novel color image steganography using Vector Quantization Codebook Generation methods LBG, TCEVR and KFCG. *In 2015 IEEE International Conference on Pervasive Computing (ICPC2015)*, 2015, January, pp. 1-6.

[9] S. Thepade, and N. Kasat. Visual content based image classification using function, rule and tree family data mining classifiers with LBG vector quantization method. *In IET International Conference & Workshop on Electronics & Telecommunication Engineering (ICWET 2016)*, 2016, February, pp. 159-162.

[10] J. Wang and N. Zheng. A novel fractal image compression scheme with block classification and sorting based on Pearson's correlation coefficient. *IEEE Transactions on image processing*, 22(9), 2013, pp. 3690-3702.

[11] W.R. Schwartz and H. Pedrini. Improved fractal image compression based on robust feature descriptors. *International Journal of Image and Graphics*, 11(04), 2011, pp.571-587.

[12] J. Wang, X. Lan, Y. Liu, N. Zheng. Fractal image encoding with flexible classification sets. *Chinese science bulletin*, 59 (14), 2013, pp. 1597-1606.