4th WSEAS International Conference on ELECTRONICS, CONTROL and SIGNAL PROCESSING, Miami, Florida, USA, 17-19 November, 2005 (pp.251-256)

An Extended LBG Algorithm for Constructing Universal Image Encoders

Chuanfeng Lv and Qiangfu Zhao
The University of Aizu
Aizuwakamatsu, Japan 965-8580

Abstract: - In recent years, principal component analysis (PCA) has attracted great attention in image compression. However, since the compressed image data include both the transformation matrix (the eigenvectors) and the transformed coefficients, PCA cannot produce the same performance as DCT (discrete Cosine transform) in respect of compression ratio. In using DCT, we need only to preserve the coefficients after transformation, because the transformation matrix is universal in the sense that it can be used to compress all images. To solve this problem we proposed k-PCA which is a potential universal image encoder. To increase the approximation ability of k-PCA, we propose an extended LBG (e-LBG) algorithm in this paper. The basic idea of the e-LBG algorithm is to improve the k-PCA iteratively using the training data. Experimental results show that the proposed approach is very effective, although the computing time is slightly increased.

Key-Words: - Image compression, PCA, vector quantization, k-PCA, universal encoder

1 Introduction

So far many techniques have been proposed for image compression. These techniques can be roughly divided into two categories: predictive approaches and transformational ones. In brief, predictive approaches like differential pulse code modulation (DPCM) and vector quantization (VQ) try to predict a pixel or a block of pixels based on known data (already observed or previously stored). Usually, only local prediction is considered. For example, in DPCM, good prediction can be made even if the predictor is very simple because neighboring pixels are often highly correlated. In VQ, a block of pixels can be predicted very well using the nearest code word.

Transformational approaches project the data into a domain which requires fewer parameters for data representation. Principal component analysis (PCA) is known as the optimal linear transformation for this purpose. Compared with VQ which approximates each point in the problem space using a different code word, PCA approximates all points using the linear combinations of the same set of basis vectors. Thus, we may consider VQ and PCA as two extreme cases. VQ is an extremely local approach which approximates each point using only one point (the nearest code word), while PCA is an extremely global approach which approximates all points using the same set of basis vectors.

So far PCA has been successfully adopted in signal processing, image processing, system control theory, communication, pattern recognition, and so on. PCA can be used to compress the dimensionality of the problem space. PCA achieves compression through discarding the principle components with small eigenvalues. However, since the compressed data must include both the transformation matrix (the eigenvectors) and the transformed coefficients, PCA cannot produce high compression ratio.

Another transformation for image compression is DCT (Discrete Cosine Transform). Although DCT is not optimal, it is one of the most popular transforms, and has been used and studied extensively. The important feature of DCT is that it takes correlated input data and concentrates its energy in just the first few transformed coefficients. The advantage of using DCT is that we need only to preserve the transformed coefficients, since the transformation matrix is universal in the sense that it can be used to compress all images.

Clearly, a PCA encoder build from one image cannot be used to compress all other images because the eigenvectors obtained from one image cannot approximate other images well. Actually, even if we consider the same image, the PCA encoder usually cannot approximate all image blocks equally well using a fixed set of eigenvector vectors. It may perform poorly in local regions which include edges or noises.

To increase the approximation ability, many improved PCA approaches have been proposed in the literature [7], [8]. The basic idea of these approaches is to train a number of PCAs which can adapt different image blocks with distinct characteristics. Though these algorithms can improve conventional

PCA in some extend, they are very time consuming and cannot be used easily.

Currently, we proposed a new approach called k-PCA which is a combination of VQ and PCA [10]. The basic idea is to divide the problem space into k clusters (sub-spaces) using VQ, and then find a set of eigenvectors using PCA for each cluster. The point is that if the training data contain enough information, we can construct a set of eigenvectors which can be used as a universal encoder to compress any input image.

Clearly, the k-PCA obtained by just combining VQ and PCA is not optimal. To improve the approximation ability of k-PCA, we propose an extended LBG (e-LBG) algorithm in this paper. Similar to the LBG algorithm, the e-LBG algorithm trains the k-PCA iteratively using the training data. Experimental results show that the e-LBG algorithm can produce much better k-PCA, with a slightly more computing time.

This paper is organized as follows: Section 2 provides a short review of VQ and PCA, and introduces briefly the concept of MPC (mixture of principle component). In Section 3, the k-PCA approach is first introduced, and then the e-LBG algorithm is proposed. The proposed algorithm is verified through experiments in Section 4. Section 5 is the conclusion.

2 Preliminaries

2.1 Vector Quantization (VQ)

VQ extends scalar quantization to higher dimensions. This extension opens up a wide range of possibilities and techniques not present in the scalar case. To implement VQ, the first step is to initialize a codebook based on the input data. The LBG algorithm as a standard approach has been widely adopted in many data compression system [1]. Its main steps are as follows:

Step 0: Select a threshold value a (>0), set k=1 and set the mean of all input data (the training data) as the first code word: $C_k^{(1)}$ (here k=1).

Step 1: If k is smaller than the pre-specified codebook size, continue; otherwise, terminate.

Step 2: Split each of the current code words into two by duplicating it with a small noise.

Step 3: Based on the current codebook, calculate the distortion, say e0. For each code word, find all the input data which satisfy:

$$d(B_m, C_i) = \min_j d(B_m, C_j) \quad (1)$$

where B_m ($m \in [1, P]$) is the m-th input datum, and P is the number of input data.

Step 4: Re-calculate each code word as the mean of the input data found in the last step. Based on the new code word, calculate the reconstructed distortion say e_1. If e_0-e_1<a then go to step 1; else go to step 3.

The distortion is often defined as the mean squared error (MSE) given by

$$MSE = \frac{1}{P}\sum_{m=1}^{P}\|B_m - C_m\|^2 \quad (2)$$

where C_m is the nearest code word for the m-th block B_m. and $\|\ \|$ is the Euclidean distance between two vectors. The distortion can also be defined as the peak signal to noise ratio (PSNR) as follows:

$$PSNR = 10\log_{10}\frac{f_{\max}^2}{MSE}\ (\text{dB}) \quad (3)$$

where $f_{\max}$ is the maximum value of the image. For a gray image with eight bits per pixel, $f_{\max}$ is 255.

After building the codebook, the coding procedure is very simple. For each input datum, find the nearest code word in the codebook. The index of the code word will be the code of this datum. For decoding, iteratively read in the index stream first, substitute each index with the code word, and put it to the image in order.

VQ is a piece-wise-linear approach. It approximates each point locally. It is locally linear but globally non-linear. It uses only one code word for each input vector. In addition, VQ is a pure discrete representation of the data, and thus can achieve high compression ratio.

There are mainly two problems in using VQ. The first one is the so called trade off relation between the compression ratio (Cr) and the fidelity. For example, in order to improve Cr, the codebook size needs to be reduced but the fidelity will be decreased. To resolve this problem, we have proposed an iterated function system (IFS) based algorithm in [2], [3]. The second shortage of VQ is the computational cost for building the codebook, which has become a bottleneck for applying VQ. This is actually one of the major research topics for improving VQ.

2.2 Principal Components Analysis (PCA)

PCA, also known as Karhunen-Loève transformation in communication theory, can maximize the decreasing rate of the variance of the input data,

through resolving the eigenvalue problem:

$$Rq = \lambda q \quad (4)$$

where R is the correlation matrix of the input data, λ is the eigenvalue of R, and q is the eigenvector. If the problem space is N-dimensional, we can have N possible solutions for the vector q. The principal components can be defined as follows:

$$a_j =< x, q_j >, \ j = 1,2,...,N \quad (5)$$

where a_j denotes the projections of x onto the j-th principal direction. To reconstruct the original data, we simply have

$$x = \sum_{j=1}^{N} a_j q_j \quad (6)$$

Usually, some of the eigenvalues are very small, and the corresponding eigenvectors can be omitted in Eq. (6). This is the basic idea for data compression based on PCA. The more eigenvectors we omit, the higher the compression ratio will be.

In 1982, Oja [3] proposed a self organized neural network with constrained Hebbian learning rule that can extract the principal component from stationary input data. Thereafter, there has been increasing interest in the study of connections between PCA and neural networks. A symmetrical multilayer perceptron (MLP) neural network [4] with the back propagation algorithm in *supervised autoassociative*, have been shown closely connected to PCA. Sanger's generalized Hebbian algorithm (GHA) [5] which extends Oja's single model to M principal components. Kung and Diamantara [6] proposed an adaptive principal component extraction (APEX) model, in which the output of the m-th principal component can be calculated based on the previous *m-1* components.

2.2 Mixture of Principle Components (MPC)

By implementing PCA we know that, it is one image vs. one transform method, since for each image we should build one particular transformation matrix consisting of eigenvectors. To reconstruct the image, not only the transformed coefficients but also the transform matrix is required. Furthermore PCA is a linear approach; it cannot approximate all areas of the image equally well. In other words, one PCA cannot simultaneously capture the features of all regions.

To resolve the above problems, MPC has been studied [7], [8]. The procedure is as follows: before PCA, divide the problem space into a number of sub-spaces, and then find a set of eigenvectors for each sub-space. If enough training data are given, MPC can construct a system which maintains a good generality. It is interesting to note that an MPC can actually be used as a universal encoder if the generalization ability is high enough. In this case, we do not have to preserve the MPC parameters in the compressed data. Only the transformed coefficients (the output of the system) for each input image block are needed.

So far researches have been focused on how to divide the problem space efficiently. In [7], Donny proposed an optimally adaptive transform coding method. It is composed of a number of GHA neural networks. Fig. 1 illustrates how the appropriate GHA is selected to learn from the current input vector. The training algorithm is as follows:

Step 1: Initialize (at random) K transformation matrices W_1, W_2, Λ, W_K, where W_j is the weight matrix of the j-th GHA network.

Step 2: For each training input vector x, classify it to the i-th sub-space, if

$$P_i x = \max_{j=1}^{K} P_j x \quad (7)$$

where $P_i = W_i^T W_i$.

Update the weights according to the following rule:

$$W_i^{new} = W_i^{old} + \alpha Z(x, W_i^{old}) \quad (8)$$

Where □ is the learning rate and Z is a GHA learning rule which converges to the principal components.

Step 3: Iteratively implement the above training procedure until the weights are stable.

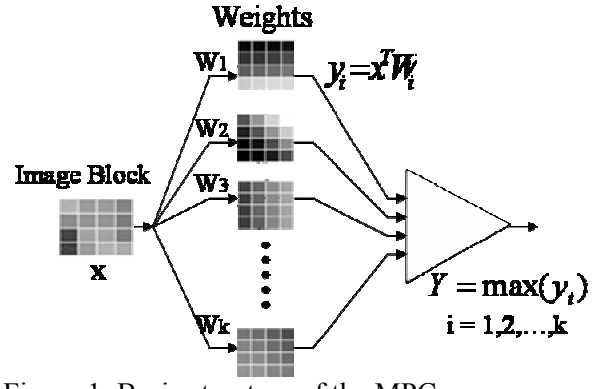

Figure 1: Basic structure of the MPC

In [7], the training parameters were given as follows: 1) the number of sub-spaces is 64 and 2) the number of training iterations is 80,000. Note that to use the MPC as a universal encoder; we must train it using many data. The above algorithm is not good enough because it is too time consuming. In paper [8], several methods were proposed to speed up the training process and decrease the distortion. These methods include growth by class insertion, growth by

components addition and tree structured network. The essential issue is that the convergence speed of GHA is very slow.

3 A New Universal Encoder: k-PCA

3.1 The concept of k-PCA

As can be seen from the pervious discussion, the computational cost of the MPC is very high. One reason is that the weight matrices to be updated are of high dimensionality, and another reason is that the convergent speed of the GHAs is slow. To solve these problems, we propose to divide the problem space using VQ. First, the dimension of the vectors (code words) to be updated is much smaller. Second, the LBG algorithm is much faster than the algorithm given in the last section. Third, for each cluster, we do not use a GHA, but a PCA, and to get a PCA is much faster. The encoding and decoding procedure of the proposed method is given in Fig.2.

Step 1: Divide the input image into $n \times n$ small blocks (n=8 here). For the entire input data, find an 8-D PCA encoder. By so doing we can reduce the dimension of the problem space from 64 to 8.

Step 2: Find a codebook with k (k=64 in our experiments) code words using the LBG algorithm, for the 8-D vectors obtained in the last step, and record the index of each input vector.

Step 3: Based on the codebook, we can divide the problem space into k clusters. For each cluster, we can find an M-D (M=4 in this paper) PCA encoder.

Step 4: For each input vector, compress it to an 8-D vector using the PCA encoder found in Step 1, then find the index of the nearest code word found in Step 2, and finally compress it to an M-D vector. The M-D vector along with the index of the nearest code word is used as the code of the input vector.

The purpose of Step 1 is to reduce the computational cost of VQ. Through experiments we have found that an 8-D PCA encoder can represent the original image very well. The codebook obtained based on the 8-D vectors performs almost the same as that obtained from the original 64-D vectors. In this paper, we call the above encoding method the k-PCA. Note that if we train the k-PCA using enough data, we can use it as a universal encoder, and do not have to include the eigenvectors into the compressed data. Thus, the compression ratio can be increased.

The reconstruction (decoding) procedure is as follows:

Step 1: Read in the codes one by one.

Step 2: Find the basis vectors for the cluster specified by the index, and transform the M-D vector back to the 8-D vector.

Step 3: Transform the 8-D vector back to $n \times n$-D vector, and put it to the image in order.

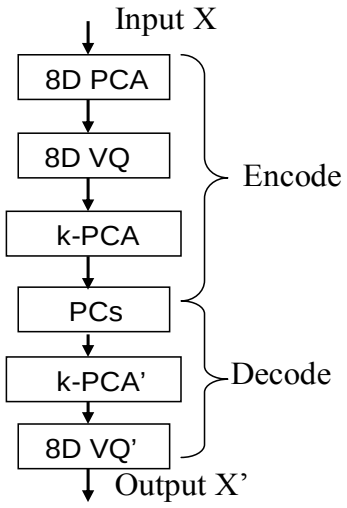

Figure 2: The flow-chat of the proposed method

3.2 The e-LBG Algorithm

From the process for constructing the k-PCA we can see that to obtain a k-PCA that generalizes well, it is important to partition the input data space properly. Although the VQ based k-PCA is better than other existing adaptive PCA approaches, it is not optimal. Actually, the decision boundaries formed by the k cluster centers are different from those formed by the k PCAs. As a result, many data cannot be encoded using the best PCA.

To improve the approximation ability of k-PCA, we can conduct a post training process based on the LBG algorithm [1]. We call this approach the extended LBG (e-LBG) in this paper. The basic idea is as follows. After partitioning the problem space by VQ, we have k set of eigenvectors. For each set of eigenvectors, we can find the set of training data that can be approximated best by the eigenvectors. The eigenvectors can be rebuilt using these data. This process can continue until the fidelity of the reconstructed image does not change significantly. This process is given as follows:

Step 1: For each input data (training data) find the closest eigenvectors which can produce smallest error between original and reconstructed data, mark it

with the index of the eigenvectors.

Step 2: For each set of eigenvectors find all input data marked by its index, using these input data construct a new set of eigenvectors and replace the old one.

Step 3: Evaluate the error between the original and reconstructed image. Compare the error with the one in last iteration.

Step 4: If there is no significant improvement, terminate; otherwise, return to Step 1.

4 Experimental Results

To verify the proposed method, we conducted experiments with ten popular images. The size of the images is the same, and is 512*512 pixels. There are 256 gray levels. So the uncompressed size of each picture is 256 kB. In the first set of experiments, we constructed the k-PCA using one image and tested the performance using the same image. The block size n is 8, the codebook size k is 64, and the number of basis vectors M is 4. Each principal component was quantized to 8bits.

The MSE (mean squared error) of the k-PCAs obtained by different methods are shown in Table.1, where "training time" is the number of iterations used by e-LBG for improving the k-PCA. From these results we can see that in this case the k-PCA after e-LBG learning is significantly better than the original one. The compression ratio in this case is 3.084 (Since the transformation matrix as well as the transformed coefficients are all counted, the compression ratio is quite low). For comparison some experimental results from [8] are also given in Table 2, which contains only results for the image Lena. In Table 1 and Table 2, the parameters n, k and M are the same. Clearly, the k-PCA is always better than the results given in Table 2.

To confirm the generalization ability of k-PCA, which is very important if we use it as a universal encoder, we conducted another set of experiments. Specifically, we used 9 of the 10 images for training, and test the resulted encoder using the remaining image. This method is often called cross-validation in machine learning. The basic idea is that if the training samples are enough, good performance for the test image can be expected.

The MSE of 5-D PCA (PCA using 5 principle components) for training and test data are given in Table 3. The MSE of k-PCA before e-LBG learning is given in Table 4. The MSE of k-PCA after 10 iterations of learning is given in Table 5, and that after convergence is given in Table 6. These results proved that the e-LBG algorithm can improve the generalization ability in call cases. Table 5 shows that the e-LBG algorithm converges very quickly. Usually, 10 iterations are enough.

Table 1: MSE of the k-PCAs for the same image

Name	VQ based k-PCA	e-LBG based k-PCA	Training time
Airport	140.2717	117.6078	28
Barbara	113.0356	79.3661	25
Boat	84.0778	66.0906	24
Elaine	37.7442	31.1575	35
F16	56.6605	43.7869	46
Lena	32.6923	25.2609	27
Man	116.6767	93.5214	28
Mandrill	331.0888	287.4656	22
Peppers	40.1849	30.7885	27
Zelda	18.7482	14.2024	33

Table 2: Results of existing methods

Results for Lena	MSE	PSNR
PCA	75.95	29.3
Growth MPC	57.1	30.06
Tree MPC	57.0	30.05
Standard MPC	84.9	28.8

Table 3: MSE of 5-D PCA (compression ratio is 12.8)

Name	Training	Test
Airport	161.0203	236.4607
Barbara	157.1405	271.8593
Boat	170.2529	154.6585
Elaine	180.8336	58.1127
F16	172.8723	129.9564
Lena	180.3825	62.4814
Man	166.4856	188.0179
Mandrill	134.1961	478.4251
Peppers	177.0497	92.9295
Zelda	183.8503	30.6501

Notice that we are trying to train a set of universal eigenvectors that are good for any image. The BPP (bit per pixel) is only calculated in terms of the transformed coefficients. From the test results, we can see that the proposed method has improved the fidelity of decoded image in all cases, with a bit increase in cost for training. Of course, if the k-PCA is constructed off-line, this additional cost can be ignored for encoding.

Table 4: MSE of k-PCA before e-LBG learning (compression ratio is 13.47)

Name	Training	Test
Airport	115.2744	175.9509
Barbara	112.6063	273.1898
Boat	122.3998	106.9265
Elaine	129.3118	49.0923
F16	124.9564	79.9465
Lena	130.3223	46.5897
Man	112.6063	133.275
Mandrill	92.0624	378.5847
Peppers	127.66	58.0644
Zelda	131.9188	23.28

Table 5: MSE of k-PCA after 10 iterations of learning

Name	Training	Test
Airport	94.1622	160.4762
Barbara	96.8948	227.2377
Boat	101.0106	98.098
Elaine	106.0526	45.3273
F16	103.817	73.1064
Lena	107.7527	40.4854
Man	98.3824	122.5518
Mandrill	73.622	361.5066
Peppers	106.4611	49.8636
Zelda	108.5873	21.7958

Table 6: MSE of k-PCA after convergence

Name	Training	Test	Times
Airport	92.9481	160.7153	74
Barbara	95.3935	224.536	68
Boat	98.6786	97.8666	110
Elaine	104.3286	44.9049	108
F16	101.5062	73.2422	75
Lena	106.1242	40.1189	81
Man	96.4446	122.4422	91
Mandrill	71.1037	361.1015	75
Peppers	105.4487	49.1948	70
Zelda	107.139	21.7178	161

5 Conclusions

In this paper we have focused our attention on how to improve the performance of previously proposed k-PCA image compression approach. An extended LBG (e-LBG) algorithm has been proposed. The basic idea of the e-LBG algorithm is to improve the k-PCA iteratively using the training data. Experimental results have shown that the proposed approach is very effective, although the computing time is slightly increased.

References:

[1] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantization," IEEE Trans. On Communications, Vol. 28, No.1, pp.84-95, 1980.

[2] C. F. Lv and Q. Zhao, "Fractal Based VQ Image Compression Algorithm," Proc. of the 66th National Convention of IPSJ, Japan, 2004.

[3] C. F. Lv, "IFS+VQ: A new method for Image Compression," Master Thesis, the University of Aizu, Japan, 2004.

[4] E. Oja, "A simplified neuron model as a principal component analyzer", J. Math. Biology 15, pp. 267-273, 1982.

[5] S. Carrato, Neural networks for image compression, Neural Networks: Adv. and Appl. 2 ed., Gelenbe Pub,North-Holland, Amsterdam, 1992, pp. 177-198.

[6] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network", Neural Networks 2, pp. 459-473, 1989.

[7] S. Y. Kung and K. I. Diamantaras, "A neural network learning algorithm for adaptive principal component extraction (APEX)", in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing 90, pp. 861-864, (Al-burqurque, NM), April 3-6 1990.

[8] R. D. Dony, "Adaptive Transform Coding of Images Using a Mixture of Principal Components". PhD thesis, McMaster University, Hamilton, Ontario, Canada, July 1995.

[9] R. D. Dony, "A Comparison of Hebbian Learning Methods for Image Compression using the Mixture of Principal Components Network" Proceedings of SPIE, v 3307, Applications of Artificial Neural Networks in Image Processing III, pp. 64-75, 1998.

[10] C. F. Lv and Q. F. Zhao, "A simplified MPC for image compression," Proc. International Conference on Computer and Information Technology, pp. 580-584, Shanghai, 2005.