A Universal PCA for Image Compression

Chuanfeng Lv and Qiangfu Zhao

The University of Aizu. Aizuwakamatsu, Japan 965-8580
{d8061105,qf-zhao} @u-aizu.ac.jp

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 performance like 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. In this paper we consider to build a universal PCA by proposing a hybrid method called k-PCA. The basic idea is to construct k sets of eigenvectors for different image blocks with distinct characteristics using some training data. The k sets of eigenvectors are then used to compress all images. Vector quantization (VQ) is adopted here to split the training data space. Experimental results show that the proposed approach, although simple, is very efficient.

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, but they are very time consuming and cannot be used easily.

In this paper, we propose a new approach named k-PCA by combining VQ and PCA. The basic idea is to divide the problem space roughly using VQ, and then find a different set of eigenvectors using PCA for each cluster (not sub-space). The point is, if the training data are complicated enough, we can construct a set of universal eigenvectors which can be used to compress any input image. Experimental results show that the proposed k-PCA approach, although simple, outperforms existing methods in the sense that the reconstructed images can have better quality without decreasing the compression ratio.

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, we propose the k-PCA approach by combining VQ and PCA. The proposed method 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: $C_k^{(1)}$ (were 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:

$$d(B_m, C_i) = \min_j d(B_m, C_j) \tag{1}$$

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

$$MSE = \frac{1}{P}\sum_{m=1}^{P}\left\|B_m - C_m\right\|^2 \tag{2}$$

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:

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

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. The k-PCA approach proposed in this paper might be one of the promising methods for solving this problem

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:

$$Rq = \lambda q \tag{4}$$

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:

$$a_j = < x, q_j >, \quad j = 1,2,...,N \tag{5}$$

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

$$x = \sum_{j=1}^{N} a_j q_j \tag{6}$$

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.

2.3 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. When reconstructing 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 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 $W_1, W_2, \cdots, W_K$, 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

$$P_i x = \max_{j=1}^{K} P_j x \tag{7}$$

where $P_i = W_i^T W_i$. Update the weights according to the following rule:

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

Where α 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.

In [7], the training parameters are: 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 clearly 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 convergent speed of GHA is very slow.

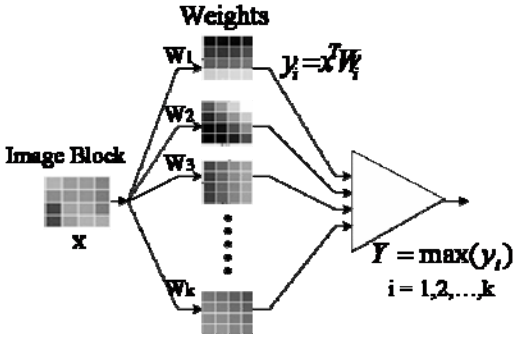

Fig. 1. Basic structure of the MPC

3. The 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.

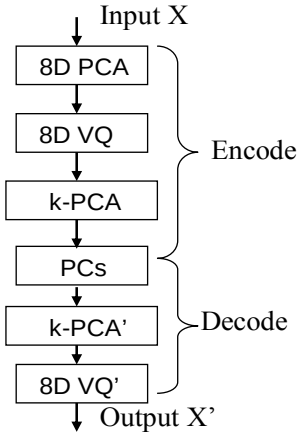

Fig. 2. The flow-chat of the proposed method

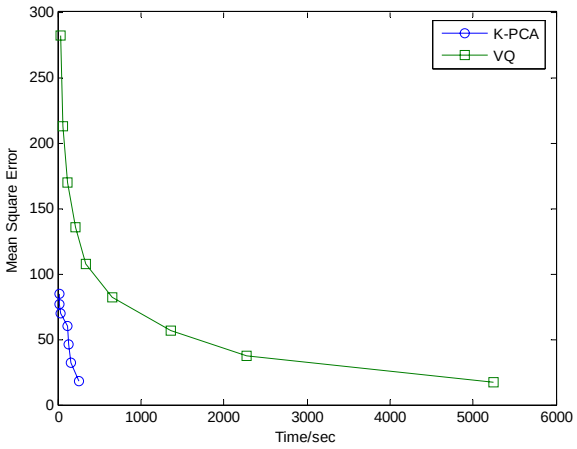

Fig. 3. Training time vs. MSE of VQ and k-PCA. The training image is Lena. Block size is 8*8; VQ is base on LBG algorithm.

4. Experimental Results

To verify the proposed method, we conducted experiments with six 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 experimental results are shown in Table.1. Here, 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 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, containing only results for the image Lena. In Table 1 and Table 2, the parameters n, k and M are the same. From the last row of Table 1 we can see that the proposed method is better than all results given in Table 2. Note that for MSE, smaller is better. For PSNR, larger is better.

In principle, VQ may achieve higher compression ratio than the proposed method. However, it is usually very time consuming. Fig. 3 shows the relation between the training time and the MSE of VQ and the k-PCA. Obviously, the computational cost for building the codebook is very high, and thus VQ cannot be used in many real-time applications. Thus, in this paper, we use VQ only for finding the k clusters, and the k-PCA is then found based on these clusters.

To confirm the generalization ability of the proposed method, we conducted another set of experiments. Specifically, we used 5 of the 6 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 training and test results PCA and the proposed method are given in Tables 3-4. Two of the reconstructed images which use the proposed method are shown in Fig. 4 and Fig. 5 respectively.

Table 1. Result of the proposed method

	MSE	PSNR
Boat	102.26	27.86
Barbara	151.92	26.32
Lena	46.03	31.50
Mandrill	350.4	22.68
Peppers	55.33	30.71
Zelda	26.42	33.91

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. Result of 5D PCA (compression ratio is 12.8)

MSE/PSNR	Training	Test
Boat	185.81/25.44	157.41/26.16
Barbara	162.05/26.03	273.63/23.76
Lena	203.81/25.04	64.91/30.00
Mandrill	120.916/27.31	479.37/21.32
Peppers	197.92/25.17	93.89/28.40
Zelda	210.26/24.90	30.90/33.23

Table 4. Result of proposed method (compression ratio is 13.47)

MSE/PSNR	Training	Test
Boat	56.96/30.57	114.49/27.54
Barbara	55.61/30.67	239.44/24.34
Lena	59.47/30.38	48.15/31.30
Mandrill	22.61/34.58	398.36/22.13
Peppers	91.05/28.53	64.856/30.01
Zelda	59.37/30.39	23.799/34.37

Notice that we are trying to train a set of universal eigenvectors that are good for any image. The BBP (bit per pixel) is only calculated in terms of the transformed coefficients. From the test results, we can see that the proposed method has better generali-

zation ability in all cases, although k-PCA has a little bit higher compression ratio. Of course, we used only five images for training the k-PCA in the experiments. If we use more images, the generalization ability can be further improved. This means that the proposed k-PCA is a very promising method for universal image compression.

Fig. 4. Reconstructed Zelda image in Table 4. The compression ratio is 13.47, the testing MSE and PSNR are 23.799, 34.37. Encoding time is 7.14 Sec, Decoding time is 0.328 Sec.

Fig. 5. Reconstructed Lena image in Table 4. The compression ratio is 13.47, the testing MSE and PSNR are 48.15, 31.30. Encoding time 7.172 Sec, Decoding time is 0.406 Sec.

5. Conclusions

In this paper we have focused our attention on how to improve the performance of PCA based image compress-ion techniques. PCA and several improved PCA methods have been compared through experiments. Based on the respective advantages and disadvantages, a hybrid approach called k-PCA has been proposed. In this method, a well trained universal eigenvectors act as a common transformation matrix like cosine function in DCT, and the VQ is used to divide the training data into k clusters. A pre-PCA has also been used to reduce the time for building the VQ codebook. We have shown that the proposed method is actually better than all existing methods in respect of reconstruction fidelity, generalization ability and computational cost.

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.