# Image compression using PCA with clustering

Chih-Wen Wang
Dept. of Inform. Eng., I-Shou University
Kaohsiung, Taiwan
gk1487@gmail.com

Jyh-Horng Jeng
Dept. of Inform. Eng., I-Shou University
Kaohsiung, Taiwan
jjeng@isu.edu.tw

*Abstract*—**Principal component analysis (PCA), a statistical processing technique, transforms the data set into a lower dimensional feature space, yet retain most of the intrinsic information content of the original data. In this paper, we apply PCA for image compression. In the PCA computation, we adopt the neural network architecture in which the synaptic weights, served as the principal components, are trained through generalized Hebbian algorithm (GHA). Moreover, we partition the training set into clusters using K-means method in order to obtain better retrieved image qualities.**

***Keywords-Image compression; K-means algorithm; Principal component analysis; Generalized Hebbian algorithm***

## I. INTRODUCTION

Clustering algorithm [1] is to partition a data set into some clusters such that data of the same cluster have some similarity properties. K-means algorithm, proposed by Mac Queen, 1967, minimizes the sum of distance from each data to its cluster center. Because of its ability to efficiently group huge data set, K-means algorithm [2-4] is a very popular method of clustering.

The aim of data reduction techniques [5] is to provide an efficient representation of the data. Such as the Karhunen-Loeve Transform (KLT), the procedure consists of mapping higher dimensional input space to a lower dimensional representation space by means of linear transformation. In principal component analysis (PCA), the KLT needs to compute the covariance matrix of input data [6] and then extract eigenvalues and corresponding eigenvectors by solving the eigen problem. The dimension reduction is achieved by using the eigenvectors with the most significant eigenvalues [7] as a new orthonormal basis. But the approach is not feasible when the dimensions of the covariance matrix become too large to be evaluated. To overcome this problem, algorithms based on neural networks are proposed.

Neural principal component analysis is firstly proposed by Oja (1982) [8-11] who uses a single neuron to extract the first principal component from the input. The Oja's rule can be viewed as the modified Hebbian (1949) rule that is the simplest unsupervised learning. To extract more than one principal component, Sanger (1989) [12] proposes the generalized Hebbian algorithm (GHA) which extracts a specified number of principal components. In this paper, we partition the training set into some clusters using K-means method and apply GHA to each of the clusters to achieve the purpose of image compression.

## II. K-MEANS ALGORITHM FOR CLUSTERING

K-means is one of the well known clustering techniques which is an iterative hill climbing algorithm. Let $S=\{x_i\}_{i=1}^{l}$, $x_i \in \Re^n$ be the data set of $l$ points of dimension $n$. Assume we require to partition the data set $S$ into $k$ clusters $C_j$, $j \in \{1,2,\cdots,k\}$. First, we choose $k$ data points $z_1, z_2, \cdots, z_k$ randomly as the initial cluster centers. Then, each of the points $x_i$ in the data set is assigned to a cluster $C_j$ if

$$\left\|x_i - z_j\right\| \le \left\|x_i - z_p\right\|,\ p = 1,2,\cdots,k \text{ and } j \ne p .$$

After all of the points are assigned to a corresponding cluster, we calculate the new cluster centers $z_1^*, z_2^*, \cdots, z_k^*$ by

$$z_j^* = \frac{1}{n_j} \sum_{x_i \in C_j} x_i,\ j = 1,2,\cdots k$$

where $n_j$ is the number of elements belonging to the cluster $C_j$. We then assign the new centers $z_j^*$ to the variables $z_j$ and the iteration proceeds. The iteration terminates subject to some proper stopping criterions.

Although K-means is one of the widely used clustering methods, it is known that the solution depends on the choice of the initial cluster centers. Therefore, several trials are required to obtain a better solution.

## III. PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) can be defined as the orthogonal projection of the given data onto a lower dimensional linear space, called the principal subspace, such that the variance of the projected data is maximized.

For a given data set $S=\{x_i\}_{i=1}^{l}$, $x_i \in \Re^n$, we consider an $m$ dimensional projection subspace where $m < n$, the optimal linear projection is defined by the $m$ eigenvectors computed from the covariance matrix of the data set corresponding to the first $m$ largest eigenvalues.

PCA provides a simple and efficient method for image compression. For instance, an image block of size $8 \times 8$ can be regarded as a 64-dimensional vector. In encoding process, the inner product of a 64 dimensional vectors and eigenvector is called a compression code for all image blocks. Moreover, a code book consists of eigenvectors. On the other hand, the summation of vectors calculated by multiplying a compression code and code word in the code book is called a predicted vector which will retrieve the image in decoded process.

## IV. GENERALIZED HEBBIAN ALGORITHM (GHA)

The GHA proposed by Sanger is an online version of the well-known Gram Schmidt orthogonal algorithm. We can view GHA as a generalization of Oja's learning rule for a special type of neural networks. A single-layered feed forward neural network, shown in Fig. 1, can be learned by using GHA. Assume the output has dimension 1. Let $x(t) \in \Re^n$ be the input with zero mean, $y(t) = w^{\mathrm{T}}(t)x(t)$ be the output of the neuron and $w(t) \in \Re^n$ be the neuron's weight at time $t$. The Oja's learning rule for updating the weights is given by

$$w(t+1) = w(t) + \eta y(t)[x(t) - y(t)w(t)]$$

where $\eta$ is the learning rate.

When the output has dimension $m$, the Oja's rule can be extended. First, we assign the learning rate $\eta$ and initialize the synaptic weights of the network, $w_{ji}$, to small random values. Then, compute

$$y_j(t) = \sum_{i=1}^{n} w_{ji}(t)x_i(t), \quad i = 1,2,\cdots,n \quad j = 1,2,\cdots,m$$

$$\Delta w_{ji}(t) = \eta y_j(t)\left( x_i(t) - \sum_{k=1}^{j} w_{ki}(t) y_k(t) \right)$$

where $x_i(t)$ is the *i*th component of the $n$ by 1 input vector $x(t)$ and $m$ is the number of principal components. Continue the iterative steps until some stopping criterions meet. However, for large $t$, the weights $w_{ji}$ of neuron *j* converges to the *i*th component of the eigenvectors associated with the *j*th eigenvalues of the covariance matrix of the input vector $x(t)$.

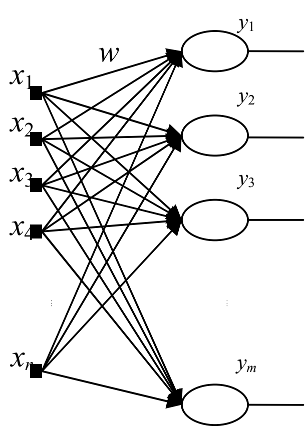



Figure 1. Single-layered feed-forward neural network for GHA

This algorithm can be rewritten in the matrix form. Let $W(t) = \{w_{ji}(t)\}$ denote the $m$ by $n$ synaptic weight matrix. Then, the update equation can be presented as:

$$y(t) = W(t)x(t)$$

$$\Delta W(t) = \eta(t)\{y(t)x^{\mathrm{T}}(t) - \mathrm{LT}[y(t)y^{\mathrm{T}}(t)]W(t)\}$$

where the operator $\mathrm{LT}[\bullet]$ makes a matrix to a lower triangular matrix which has zeros at upper-right entries.

## V. CLUSTERING FOR PCA AND IMAGE CODING

In this study, we partition the training set into $k$ clusters using K-means method and apply PCA to each of the clusters. To obtain the principal eigenvectors, we employ GHA to train the weights which serve as the principal components. Then we can use these components to encode and decode the input images.

The purpose of clustering analysis is to classify objects into subsets that have some meaning in the context of a particular problem. More specifically, the training data in an $n$ dimensional space are grouped into $k$ clusters in such a way that patterns in the same cluster are similar in some sense. Here, to obtain better retrieved image qualities, K-means algorithm is adopted.

PCA achieves the purpose of image compression by replacing $n$ dimensional vectors, with $m$ dimensional outputs where $m < n$. The smaller is the value of $m$, the greater is the compression ratio. We can use the amount of weights to compare the compression ratio for different $k$ clusters. For example, a training set $S_k = \{x_i\}_{i=1}^{r}$, $x_i \in \Re^n$, consists of non-overlapping $8 \times 8$ image blocks where $r$ is the number of data points for one cluster. If there are $m$ output nodes in the neural network with $k$ clusters, the total numbers of weights are $n \times m \times k$. But PCA is not efficient when the dimension of the covariance matrix becomes too large. To overcome this problem, GHA is used for the network training.

For practical implementation, we propose a selecting mechanism to balance the trade-off between the compression ratio and the reconstructed image quality.

## VI. EXPERIMENTAL RESULTS

Simulation programs are implemented by using Borland C++ Builder 6.0 running on Microsoft Windows XP, Intel Core 2 Duo 1.8-GHz CPU, and 1.96-GB RAM platform. We test the results of the K-means and GHA by examining its use for image compression.

In PCA, the training images are Lena and F16 of size $256 \times 256$, which are shown in Fig. 2 and Fig. 3 respectively. The training set $S = \{x_i\}_{i=1}^{l}$, $x_i \in \Re^n$ consists of non-overlapping $8 \times 8$ image blocks, i.e., $n = 64$ and $l = 1024$. In the training process, we set the learning rate as $\eta = 10^{-3}$ and a fixed iteration number of 20,000.

The similarity between two image blocks $u$ and $v$ of the same size $L \times L$ is measured in terms of the MSE (mean squared error) defined as follows:

$$\mathrm{MSE}(u,\, v) = \frac{1}{L^2} \sum_{i,j=0}^{L-1} [u(i,\, j) - v(i,\, j)]^2 ,$$

and the distortion between the original image $f$ and the retrieved image $\hat{f}$ caused by lossy compression is measured in peak signal to noise ratio (PSNR) defined by

$$\mathrm{PSNR}\left(f, \hat{f}\right) := 10 \cdot \log_{10}\left(\frac{255^2}{\mathrm{MSE}\left(f, \hat{f}\right)}\right).$$

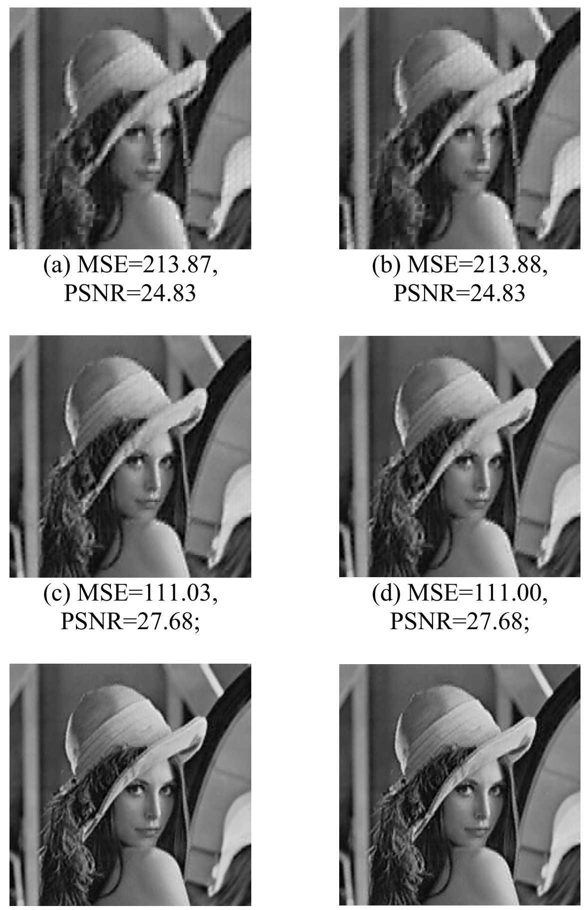
We compare the quality of reconstructed images between the GHA method and direct computation of the first $m$ principal components from the covariance matrix for $m = 4, 8, 16$. Results shown in Fig. 4 and Fig. 5 indicate that more extracted components exhibits better visual effects.

For clustering study, we partition the training set into 10 clusters using K-means. In the network, the number of output nodes is set as $m = 16$. For each of clusters, PCA is performed separately. The MSE and PSNR evaluated inside each cluster using the trained weights are shown in Table 1 and Table 2. The retrieved images are shown in Fig. 6 and Fig. 7. It is clear that the reconstructed images have very good qualities. We have MSE=25.89 and PSNR=34.00 dB for Lena and MSE=53.59 and PSNR=30.84 dB for F16. However, for this case, there are 10,240 weights required to record.

For 5-cluster case the results are shown in Table 3 and Table 4. The retrieved images are shown in Fig. 8 and Fig. 9 for Lena with MSE=37.42 and PSNR=32.40 dB and F16 with MSE=81.86 and PSNR=29.00 dB. Although the quality decays, fewer weights are required to record where the total number of used weights is 5,120.

Figure 2. Training image (Lena)


Figure 3. Training image (F16)



(a) MSE=213.87, PSNR=24.83 (b) MSE=213.88, PSNR=24.83

(c) MSE=111.03, PSNR=27.68; (d) MSE=111.00, PSNR=27.68;

(e) MSE=50.48, PSNR=31.10 (f) MSE=56.65, PSNR=30.60

Figure 4. Image (Lena) sequence obtained with eigenvectors as (a)(c)(e) and final weights as (b)(d)(f).

(a) MSE=389.00, PSNR=22.23

(b) MSE=390.14, PSNR=22.22

(c) MSE=230.79, PSNR=24.50;

(d) MSE=232.24, PSNR=24.47;

(e) MSE=114.74, PSNR=27.53

(f) MSE=116.49, PSNR=27.47

Figure 5. Image (F16) sequence obtained with eigenvectors as (a)(c)(e) and final weights as (b)(d)(f).

TABLE I. Comparisons of training 10 clusters (Lena)

| No. Cluster | # Blocks | MSE | PSNR |
|---|---|---|---|
| 1 | 32 | 13.5 | 36.83 |
| 2 | 58 | 13.3 | 36.89 |
| 3 | 64 | 83.2 | 28.93 |
| 4 | 150 | 24.9 | 34.17 |
| 5 | 149 | 5.9 | 40.42 |
| 6 | 29 | 11.0 | 37.72 |
| 7 | 160 | 40.2 | 32.09 |
| 8 | 114 | 12.7 | 37.09 |
| 9 | 165 | 35.2 | 32.67 |
| 10 | 103 | 11.6 | 37.49 |

TABLE II. Comparisons of training 10 clusters (F16)

| No. Cluster | # Blocks | MSE | PSNR |
|---|---|---|---|
| 1 | 43 | 148.93 | 26.40 |
| 2 | 65 | 146.67 | 26.47 |
| 3 | 27 | 20.03 | 35.11 |
| 4 | 20 | 18.86 | 35.38 |
| 5 | 21 | 21.30 | 34.85 |
| 6 | 263 | 35.24 | 32.66 |
| 7 | 64 | 131.65 | 26.94 |
| 8 | 386 | 8.75 | 38.71 |
| 9 | 75 | 142.40 | 26.60 |
| 10 | 60 | 97.87 | 28.22 |

Figure 6. Image (Lena) obtained with all 10 clusters

Figure 7. Image (F16) obtained with all 10 clusters

TABLE III. Comparison on training 5 clusters (Lena)

| No. Cluster | # Blocks | MSE | PSNR |
|---|---|---|---|
| 1 | 114 | 14.43 | 36.54 |
| 2 | 214 | 53.58 | 30.84 |
| 3 | 269 | 24.48 | 34.24 |
| 4 | 220 | 34.64 | 32.74 |
| 5 | 207 | 53.34 | 30.86 |

TABLE IV. Comparison on training 5 clusters (F16)

| No. Cluster | # Blocks | MSE | PSNR |
|---|---|---|---|
| 1 | 98 | 203.41 | 25.05 |
| 2 | 116 | 181.10 | 25.55 |
| 3 | 95 | 194.52 | 25.24 |
| 4 | 460 | 8.92 | 38.63 |
| 5 | 255 | 79.28 | 29.14 |

Figure 8. Image (Lena) reconstructed by all of 5 clusters


Figure 9. Image (F16) reconstructed by all of 5 clusters

## ACKNOWLEDGMENT

This work has been supported by the National Science Council of Taiwan, under grants NSC 99-2221-E-214-053-MY2.

## REFERENCES

[1] Murat Erisoglu, Nazif Calis, and Sadullah Sakallioglu, "A new algorithm for initial cluster centers in k-means algorithm," Elsevier Science, *Pattern Recognition Letters* 32 (2011) 1701-1705.

[2] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, Reading, MA, 1974.

[3] S.Z. Selim and M.A. Ismail, "K-means type algorithms: a generalized convergence theorem and characterization of local optimality," *IEEE Trans. Pattern Anal. Mach. Inteli* 6 (1984) 81−87.

[4] H. Spath, *Cluster Analysis Algorithms*, Ellis Horwood, Chichester, UK, 1989.

[5] S. Costa and S. "Fiori, Image compression using principal component neural networks," Elsevier Science, *Image and Vision Computing* 19 (2001) 649-668.

[6] L. Xu, "Least mean square error reconstruction principle for self-organizing neural-nets," *Neural Networks* 6 (1993) 627-648.

[7] K.I. Diamantaras and S.Y. Kung, *Principal Component Neural Networks: Theory and Applications*, Wiley, New York, 1996.

[8] E. Oja, "A simplified neuron model as a principle component analysis," *J. Math. Biol*. 15 (1982) 267-273.

[9] E. Oja, "Neural networks, principle components and subspaces," *Int J. Neural System* 1 (1989) 61-68.

[10] E. Oja, "principle components, minor components, and linear neural networks," *Neural networks* 5 (1992) 927-935.

[11] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biol*. 15 (1982) 267-273.

[12] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feed-forward neural network," *Neural Networks* 2 (1989) 459-473.-441, 498-520.