Implementation of 2D-PCA

Pushkar Desaraju Sidhant Agarwal

***Abstract*—In this paper, we will be discussing the results obtained after implementation of 2-dimensional Principal Component Analysis based on the paper Two-dimensional PCA: a new approach to appearance-based face representation and recognition". The goal of this project is to understand the algorithm proposed in the paper by Zhang et.al [1] and compare the results with traditional PCA. We will be discussing the results of our implementation using MATLAB while using ORL database.**

I. Introduction

PRINCIPAL Component Analysis is a method to extract features and highlight the differences and the similarities of the images in a given dataset. To implement PCA algorithm on image dataset, we first need to transform the input image matrix into 1D column vectors. This transformation would increase the dimensionality of the covariance matrix and increase the cost of computations. Therefore it is difficult to calculate the covariance matrix, eigenvectors accurately because of the small number of test data sample and large covariance matrix. To tackle this problem, the paper by Zhang et.al [1] suggests that we use the 2D image directly to calculate the covariance matrix.

According to the paper the resultant 2D-PCA algorithm is computationally faster than PCA because of the smaller covariance matrix. Additionally, 2D-PCA also has better recognition accuracy than PCA. Algorithms like 2D-PCA can be used in various applications like access control fraud detection and face identification. In further sections, Section II, Section III, Section IV, Section V we will be discussing the methodology used by the original authors, our understanding and implementation using MATLAB language, the dataset used for experiments and analysis of 2DPCA while performing the ORL experiment and comparison of various features of 2DPCA and PCA. In the later sections, we will be discussing the workload, acknowledgement of learning, conclusion and the references used while implementing and learning various concepts involved in the implementation.

II. Methodology Used

In this paper, the implementation is discussed in 4 sections, namely, Idea and Algorithm, Feature Extraction, Classification Method and finally Image Reconstruction. The main idea of the proposed algorithm was to take an input image as matrix A of m rows and n columns and project it onto a matrix X, which is an n-dimensional unitary column vector. The resultant matrix is Y, which is called the feature vector of image A which contains the principal components of the input.

In order to find a good projection X for the image A, the total scatter of input was introduced as trace of the covariance matrix, S_x, of the projected feature vectors. From here J(X) is computed as trace of S_x which is the sum of all elements in the principle diagonal of the matrix.

$$J(X) = tr(S_x) \tag{1}$$

An optimal projection vector X would be the one which maximizes the scatter of the resulting projected sample images. Further, the covariance matrix is defined as

$$S_x = E(Y - EY)(Y - EY)^T \tag{2}$$

In the above equation, Y was replaced with AX, therefore the equation became,

$$S_x = E[AX - E(AX)][AX - E(AX)]^T \tag{3}$$

Further, this equation can be decomposed into

$$S_x = E[(A - EA)X][(A - EA)X]^T \tag{4}$$

Now, we can get the trace of the S_x as,

$$tr(S_x) = X_t[E(A - EA)^T(A - EA)]X \tag{5}$$

From this trace equation, we take G_t, image covariance matrix as,

$$G_t = E(A - EA)^T(A - EA) \tag{6}$$

To evaluate this scatter matrix from the image dataset, the author calculated the mean of all images and denoted the value as $\overline{A}$. Using the $\overline{A}$, G_t was calculated using this equation,

$$G_t = \frac{1}{M}\sum_{j=1}^{M}(A_j - \overline{A})^T(A_j - \overline{A}) \tag{7}$$

Here, the mean is subtracted as we intend to capture only unique features of the face. From this matrix, G_t, eigenvectors, X_1, X_2, X_3, ..., X_d are extracted such that they maximize J(X). It is noted that only the maximum X is not enough, therefore total d eigenvectors are extracted. All these eigenvectors follow the orthonormal constraints and are essential part for procedure of image reconstruction.

Using the above d eigenvectors, feature extraction is performed for all the images in the dataset. The image dataset is projected upon these eigenvectors to obtain the family of projected vectors Y_1, Y_2, Y_3, ..., Y_k . As compared to the scalars in PCA, the principal components are vectors in 2D PCA. Using these features, a nearest neighbor classifier is used for classification, and the distance is calculated using Eq 8

$$d(B_i, B_j) = \sum_{k=1}^{d} || \ Y_k^{(i)} - Y_k^{(j)} \ ||_2 \tag{8}$$

TA: Kevin Ding

For image reconstruction, top principal components and eigenvectors are combined to get the original image. The combination is done using this equation:

$$\overline{A} = VU^T = \sum_{k=1}^{d} Y_k X_k^T \tag{9}$$

Here d is the selected number of eigenvectors and n is the total number of eigenvectors of G_t. Also, $\overline{A}$ represents the reconstructed image, which has the same size as that of

the original image. If d $<$ n then the reconstructed image is as approximation of the original image as not all the features and eigenvectors have been combined. Therefore to get the original image A, all the values of d were used.

III. Our Implementation

There are 3 experiments performed to analyze 2DPCA and compare it with other methodologies used for image recognition. 3 well known image datasets were used, ORL, AR and Yale. We performed the ORL experiment using ORL Dataset and analyzed the accuracy, performance and intricacies of 2DPCA vs PCA.

A. Constructing Image Covariance matrix

As discussed in paper, in 2D-PCA, we work on 2D image matrices. To construct the image covariance matrix, we used a dataset and loaded it into an array which came out to be a 3 dimensional array in MATLAB. Mean of all the training set was calculated to be used in further calculations. We then used equation 7 from the paper to calculate the G_t matrix i.e. the covariance matrix. Our understanding of this part was that since we needed only the unique identifiable features from the images, the mean was subtracted from all the image dataset and then the resultant matrices were used for the calculation.

B. Extracting d Eigenvectors

Using this new Covariance Matrix, G_t, we extracted the eigenvectors along with the corresponding eigenvalues. These eigenvectors were found to be in random order. The eigenvectors and eigenvalues were then sorted in decreasing order to get a matrix which had X_1, X_2, X_3,..., X_d as the eigenvectors. By reverse sorting we were able to choose the top d eigenvectors for feature extraction.

C. Extracting top Principal Components

These eigenvectors were then used to calculate the principal components. This was done by projecting the original image dataset onto the top eigenvectors.We multiplied the image matrix with the matrix containing the top eigen vectors and stored the result into a matrix.The resultant matrix contains the projections of input images along the principle components.It is noteworthy to note because 2DPCA's counterpart PCA has scalar Principal Components.

D. Reconstructing Images

For image reconstruction the principal components are used as described in Image Reconstruction section of paper. The process is similar to that of Eigenfaces. We take the Principal Components extracted in the previous step and compute a transpose of Eigenvector matrix. Now, we calculated the summation of multiplication of both matrix, as the column values and row value of transposed Eigenvector matrix is same. The dimensions of the reconstructed images will be similar to the input matrix

E. Classification

We used Nearest Neighbor technique to classify the images. Test dataset was created by using an image from each class to classify. Distance of a sample in the test dataset with the projected image matrix is evaluated and nearest neighbor is determined. Accuracy of the model is evaluated by checking whether the nearest neighbor and the test image belonged to the same class.

F. Implementation of PCA

We also implemented the PCA algorithm to compare its efficiency and runtime with 2DPCA. We took the same dataset and created the input data by converting each image into column vector and we used matlab functions to compute covariance matrix and eigen vectors. The code used for projections and reconstruction are modified to match dimensions of input. Also, to get the most accurate results, the experiments which required calculation of time, accuracy were run at least 4-5 times and then a mean was taken to reach a final value.

G. Assumptions

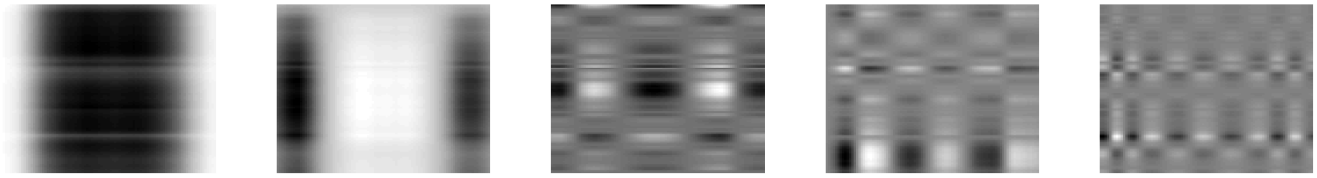
We assume that correct path to dataset will be provided by the user during testing and user will not modify directory structure of the ORL dataset We assumed that dataset will be sufficiently large enough for us to project it into k components and all necessary packages like Eig package will be installed in MATLAB before testing.

H. Potential Problems

The paper was not clear enough in explaining how covariance matrix was calculated and has presented ambiguous equations The paper has not clearly discussed the method to extract principal components from covariance matrix

IV. DATASET

In this section we discuss about the dataset on which we performed our experiment for analysis. Mostly the text of this section is taken from [3]. We performed the analysis of 2DPCA on ORL dataset i.e. Our Database of Faces, (formerly 'The ORL Database of Faces'), contains a set of face images taken between April 1992 and April 1994 at the ATT lab. The dataset contains 40 classes and 10 images are present for each person.The images are taken at varying angles with changing the lighting and facial expression of the person and facial



k=1 k=2 k=4 k=7 k=10

Fig. 1: Some reconstructed sub-images(partial) are shown in inverse color.

details like with and without glasses. The files are in .PGM format, and softwares like GIMP can be used for visualization. The size of each image is 92x112 pixels, with 256 grey levels per pixel. The images are organised in 40 directories (one for each subject), which have names of the form sX, where X indicates the subject number (between 1 and 40). In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).[3]

There are 3 experiments performed to analyze 2D PCA and compare it with other methodologies used for image recognition. 3 well known image datasets were used, ORL, AR and Yale. We performed the ORL experiment by using first 9 images in each class as training data and the last image is used for testing.

V. ANALYSIS

In the following section, we have described the analysis of our implementation of 2DPCA, and the experiment performed. As mentioned before we performed the ORL experiment described in the paper, and the dataset has already been described in section IV. We analyzed various factors in 2DPCA and in the subsequent subsections we have discussed various results and 2DPCA algorithm's comparison with PCA algorithm.

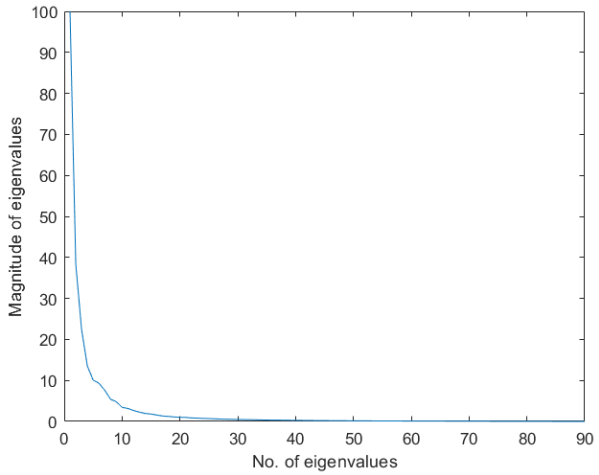


Fig. 2: The plot of the magnitude of first 90 eigenvalues in decreasing order

A. Analysis of Eigenvectors

To analyze the complexity and dimensions of the Covariance matrix, we used equation (7) to get the result. For dataset containing images for 40 subjects, taking first 9 images from each subject, gave us a covariance matrix of dimensions 92x92 in 2D-PCA as opposed to a dimension of 360x360 in PCA. The better performance of 2DPCA can be attributed to the difference in dimensions of covariance matrix. In order to extract eigenvectors, the datatype of values obtained in Covariance Matrix needs to be converted into 'double' values. To find the eigenvectors and eigenvalues we used *eig* [2] function of MATLAB. This gave us the matrix containing eigenvectors in random order. As per the theory, we needed the top eigenvectors for better result. The resultant eigenvectors were then sorted in descending order using dsort function, which sorts the complex eigenvalues in the vector in descending order by magnitude. To get a visualization of the eigenvalues whose values were normalized between 1-100 Fig 2 was plotted with the top 90 eigenvalues in descending order. We can easily see that first few vectors have the highest values, i.e. contains most of the information of the image.

B. Analysis of Principal Vectors w.r.t Partial Image Reconstruction

To observe that the eigenvectors with highest eigenvalues gave better image information, we displayed the partially reconstructed images from these top eigenvectors. In Fig 1 we can see the images in inverse color for more clarity. As the value of k is increasing we can observe that the image becomes smoother as Most of the energy is present in the first few reconstructed images, i.e. most of the features are present in those reconstructed images for which the highest eigenvectors were used. As value of k increases the information contained in the individual reconstructed images decreases. Therefore, as described in the paper, we can conclude that the energy of an image is concentrated on its first small number of component vectors. Another thing to note for k=1 and k=2, we get the orthonormal information from these 2 eigenvectors.

C. Analysis of 2DPCA vs. PCA - A Comparison

1) Theory: To understand the performance of 2D-PCA, we also implemented PCA algorithm. The main difference between PCA and 2D PCA are, in PCA we convert the 2D image l x w into a column matrix lw x 1 . We obtain the matrix lw x n where n is number of images. Then, the matrix

Fig. 3: Some reconstructed images based on 2DPCA (upper) and PCA (lower).

lw x *n* is projected into *lw* x *k* where *k* is the number of principal components. On the other hand, in 2D PCA we use the 2D image directly without converting into a column vector.

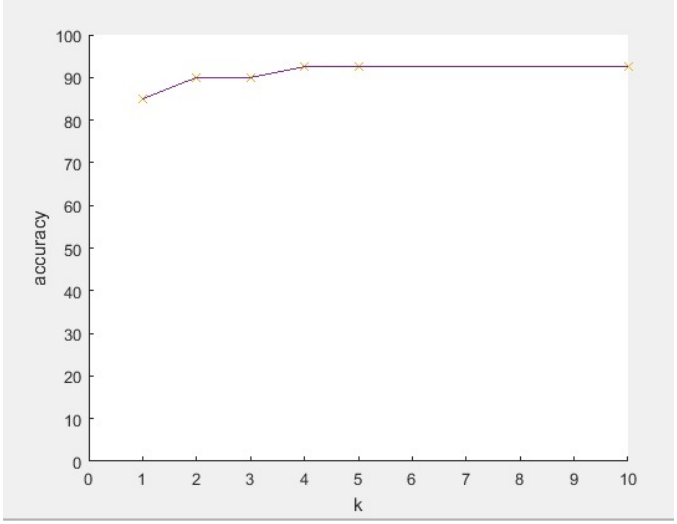
Because of this approach of the PCA algorithm, this transformation has lead to the image into higher dimensional vector space.

2) Comparison of principal components 'k' vs accuracy: From the figure 4, we can infer that 2D-PCA gives a good performance at 'k' values as low as 1. With increase in value of 'k' the performance slightly improves and remains constant after a particular value. In case of PCA, we can see that performance is very low at lower values of k and accuracy increases significantly by increasing 'k'. The value of 'k' normalizes at a value which is much higher than the value required in case of 2DPCA.This implies that 2DPCA can achieve the performance offered by PCA with fewer computations. By analyzing the two graphs we can conclude that efficiency of 2DPCA is much higher than efficiency of PCA as it can recognize faces accurately at a very low value of k.

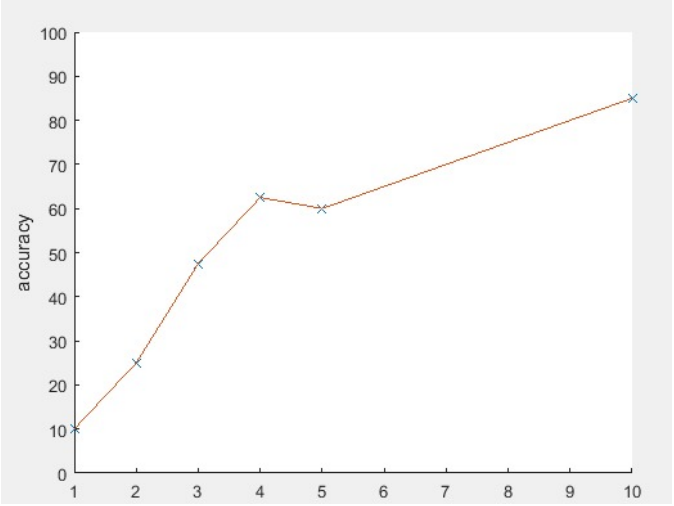
3) Comparison based on Reconstruction of Images: Both PCA and 2DPCA algorithms were used to recreate the image with same number of eigenvectors. The result of this experiment was as expected. The images constructed using 2DPCA are more clear as compared to the images constructed using PCA, taking same eigenvectors each time. We are able to visualize the difference easily in Fig 3. While increasing the number of eigenvectors in 2DPCA shows a huge difference in clarity while PCA requires a large number of eigenvectors to get the same level of clarity.

4) Comparison based on Time: We compared the feature extraction time for both algorithm and noted the result into Table I. As we can see, for every number of sample classes, 2D PCA came out to be superior than PCA. 2DPCA extracts features a lot faster than its counterpart, i.e. PCA. For this experiment we have tabulated our result in Table I. This is supported by the fact that PCA's covariance matrix is much larger in size as compared to that of 2DPCA.

5) Comparison based on Accuracy: To compare the accuracy of image recognition between PCA and 2DPCA, both algorithms were applied on ORL Database. All the required information including Covariance Matrix, Eigenvectors, Eigenfaces were extracted. Top 15 eigenvectors were taken for the both PCA and 2DPCA for a better comparison and Table II has been plotted. We can clearly see for the same number of eigenvectors 2DPCA outperforms PCA. For this comparison we chose 2, 4, 5, and 9 classes to calculate the recognition rate.



2DPCA



PCA

Fig. 4: Comparison of Accuracy between PCA and 2DPCA

TABLE I: Comparison of CPU Time (s) for Feature Extraction Using the ORL (CPU: Intel(R) Core(TM) i5-6200 @ 2.30 GHz, RAM: 8 GB)

#Training class	1	2	3	4
2DPCA	0.0310	0.0630	0.0940	0.1090
PCA	0.291502	0.356147	0.414663	0.447242

VI. Workload

We started the project by going through the theory for PCA and took help in understanding the basic concepts which were covered in class. After understanding the concepts of 2DPCA individually, we sat together and discussed the problems which we were facing. We both met with the Teaching Assistant Mr. Kevin Ding to seek his input on some mathematical equations

TABLE II: The below table is a comparison of accuracies of 2D PCA and PCA. Both of these accuracies were calculated at k=15, where k denotes the number of topmost eigenvectors obtained from Covariance Matrix

#Training samples / class	2	4	5	9
2DPCA	72%	83%	87.5%	95%
PCA	65%	75%	82.5%	87%

discussed in the paper. Pushkar has developed the code to input the dataset, reconstruct the image from projection and function for classification using Nearest Neighbor algorithm and find the accuracy. Sidhant has developed the code for calculation of covariance matrix, project the input image to 'k' components, display images, tables, plot other graphs used for comparison analysis. We both contributed in writing the mid term reports as well as writing the final report using LaTeX.

VII. Discussion

By implementing the algorithm for 2DPCA and comparing its results with PCA we found out that 2DPCA has greater recognition accuracy and it can recognize faces at a much lower value of k compared to PCA. We are basing this claim on the quality of reconstructed images, at the same value of 'k' 2DPCA produces images with much higher quality.Therefore, we can say that 2DPCA takes only a fraction of computational power and run time required for PCA to achieve the same accuracy.

VIII. Conclusion and Future scope

2DPCA is an improvement over traditional PCA because feature extraction is easier in 2DPCA as it uses the 2D matrix directly. 2DPCA is is computationally faster than PCA and it is more accurate. Additionally, by visualizing and comparing the reconstructed matrices of 2DPCA and PCA we can say that 2DPCA is better than PCA at capturing features of a face for the same value of k.

There are few areas in which 2DPCA can be further optimized.According to the paper written by Zhang et.al [1], 2DPCA algorithm requires more coefficients than PCA.As future study, we can attempt to reduce this to make it more efficient.

IX. Acknowledgement

All the concepts used in the project starting from 2DPCA implementation to the intricacies of Image manipulations were new to most of our project team members. An exceptional appreciation to our course instructor and guide Prof. Baoxin Li whose encouragement helped us to attempt the assignments, project and conveying the task effectively. Additionally, much obliged to the course TA Mr. Kevin Ding who put his opportunity in controlling our group to accomplish our objective and enable us to comprehend the ideas better.

References

[1] Zhang, Daoqiang, and Zhi-Hua Zhou. Two-Dimensional PCA: Two-Directional two-Dimensional PCA for efficient face representation and recognition. *Neurocomputing* , vol. 69, no. 1-3, 2005, pp. 224231., doi:10.1016/j.neucom.2005.06.004.

[2] Eigenvalues and eigenvectors of symbolic matrix - MATLAB eig, www.mathworks.com/help/symbolic/eig.html.

[3] The Database of Faces http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html