Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE488 Intro. to Machine Learning, Spring 2018

Issued: Apr. 30, 2018
Due: May. 23, 2018

Assignment III - part II

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take considerable amount of time so start early.

1. **PCA (Matlab Programming Assignment)**: This problem involves implementing a PCA (Principal Component Analysis) based algorithm to represent a human face in terms of principal components, and based on this representation, identify the face in a given image as one of identity in the training set. A dataset comprising of 213 human-face images of 10 individuals is provided of which 163 images are used for training and 50 are used for testing. A balanced training data is used: there are equal number of images for each identity. Each face image portrays an individual with different expression.

   (i) Perform PCA on the 163 training images with different number of principal components (PC) then display the top 5 principal components. Also, reconstruct each image using 5, 50, 200 and 500 principal components. Compare the average reconstruction mean square error versus the number of PCs.

   (ii) Use PCA representation to identify the 50 test face images. Write a Matlab code to identify input test images. Use the Euclidean distance as a measure of closeness.

   Algorithm 1: PCA algorithm

   1. Construct data matrix $\boldsymbol{X}$
   2. Subtract mean face $\bar{\boldsymbol{x}}$ from each image
   3. Construct Covariance Sample matrix $\boldsymbol{\Sigma}$
   4. Find eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$
   5. Find principal Components, which are k eigenvectors with largest eigenvalues.

   For implementing the PCA algorithm, a skeleton code is provided in the "PCA" folder. The main function `PCA_main.m` calls the following 5 functions:
   (1) `[train_matrix,test_matrix] = createDataset()`,
   (2) `[project_train_img,k_eig_vec,m] = train_PCA(train_matrix,k)`,
   (3) `[recon_error] = train_recon(train_matrix,project_train_img,k_eig_vec,m)`,
   (4) `[project_test_img] = test_PCA(test_matrix,k_eig_vec,m)`,
   (5) `[id] = identify(project_train_img,project_test_img)`

   (i) `[train_matrix,test_matrix] = createDataset()` constructs both the training and test matrix in form of $N \times d$ where N is the number of images, and d is the size of vector image. training and test image sets are contained respectively in 'training_img' and 'test_img' folder.

```
function [train_matrix,test_matrix] = createDataset()
%% Arguments %%
% N_train is the number of training dataset
% N_test is the number of training dataset
% d is the dimension of training and test dataset
% train_matrix is the training data matrix
% test_matrix is the test data matrix

%% Codes %%
addpath ./training_img
addpath ./test_img

N_train = 163;
d = 64*64;
train_matrix = zeros(N_train, d);
for k=1:N_train
fname = sprintf('training_img/%d.jpg',k);
img = double(imread(fname));
train_matrix(k,:) = (img(:))';
end


N_test = 50;
test_matrix = zeros(N_test, d);
for k=1:N_test
fname = sprintf('test_img/%d.jpg',k);
img = double(imread(fname));
test_matrix(k,:) = (img(:))';
end

end
```

(ii) `[project_train_img,k_eig_vec,m] = train_PCA(train_matrix,k)` takes training matrix `train_matrix` and parameter `k`, which is the number of PCs. This function outputs 3 values, `project_train_img,k_eig_vec,m` and display the 3 largest eigen faces, which interpret the eigen vectors as image.
`project_train_img` is the training matrix represented by k PCs. `k_eig_vec` is k PCs and `m` is mean.

```
function [project_train_img, k_eig_vec, m] = train_PCA(train_matrix,k)

%% Your code here %%
%%%%% calculating mean image vector %%%%%

m = mean(train_matrix,1);
imgcount = size(train_matrix,1);

%%%%%  calculating A matrix, i.e. after subtraction of all image vectors from the mean image vector %%%

A = [];
for i=1 : imgcount
temp = double(train_matrix(i,:)) - m;
A = [A;temp];
end

L= A' * A;
[V,D]=eig(L);  %% V : eigenvector matrix  D : eigenvalue matrix
k_eig_vec = V(:, end-k+1:end);

%%% Project the training matrix %%%
project_train_img = A * k_eig_vec;
```

```
%% display 3 biggest eigen vectors
for i = 1:3
subplot(1,3,i)
imagesc(reshape(k_eig_vec(:,end-i+1), 64, 64));
end

end
```

(iii) `[recon_error] = train_recon(train_matrix,project_train_img,k_eig_vec,m)` takes 4 inputs: training matrix `train_matrix`, projected training matrix `project_train_img`, k principal components `k_eig_vec` and mean `m`. This function saves the reconstructed training images from the training matrix represented by k eigenvectors and outputs the average reconstruction mean square error.

```
function recon_error = train_recon(train_matrix,project_train_img,k_eig_vec,m)

%% Arguments %%
% train_matrix : training data matrix with dimension of N*d
%                (N is the number of training images and d is the dimension of one images.)
% project_train_img : training matrix represented by k PCs
% k_eig_vec : k biggest eigen vectors.
% m: mean from training matrix.

%% Your code here %%
% write code to find reconstructed image 'recon_img' and reconstruction
% error.
mean = repmat(m,size(project_train_img,1),1);
recon_img = project_train_img*k_eig_vec' + mean;
recon_error = norm(train_matrix-recon_img);
%% save reconstructed images in the folder 'train_reconstruction'
face = zeros(64,64);
mkdir('train_reconstruction')
for i=1:size(recon_img,1)
fname = sprintf('train_reconstruction/%dres.jpg',i);
face(:) = recon_img(i,:);
imwrite(uint8(face), fname);
end
end
```

(iv) `[project_test_img] = test_PCA(test_matrix,k_eig_vec,m)` takes test matrix `test_matrix`, k principal components `k_eig_vec` and mean `m` and outputs `project_test_img`, the test matrix represented by k PCs from training matrix.

```
function [project_test_img] = test_PCA(test_matrix,k_eig_vec,m)

%% Your code here %%
%%%%% extractiing PCA features of the test image %%%%%
imgcount = size(test_matrix,1);
A = [];
for i=1 : imgcount
temp = double(test_matrix(i,:)) - m;
A = [A;temp];
end

%%% finally the eigenfaces %%%
project_test_img = A * k_eig_vec;

end
```

(v) `[id] = identify(project_train_img,project_test_img)` takes training matrix and test matrix both represented by k PCs as inputs and outputs the index(e.g. 1,2,...) of training images which is most similar to the test images.

```
function [recognized_img] = identify(project_train_img,project_test_img)

%% Your code here %%
%%%%% calculating & comparing the euclidian distance of all projected trained images from the projected

euclide_dist = zeros(size(project_test_img,1),size(project_train_img,1));
for i=1:size(project_test_img,1)
for j=1 : size(project_train_img,1)
temp = (norm(project_test_img(i,:)-project_train_img(j,:)))^2;
euclide_dist(i,j)=temp;
end
end

recognized_img = [];
for i=1:size(euclide_dist,1)
[euclide_dist_min, recognized_index] = min(euclide_dist(i,:));
recognized_img = [recognized_img recognized_index];
end


end
```

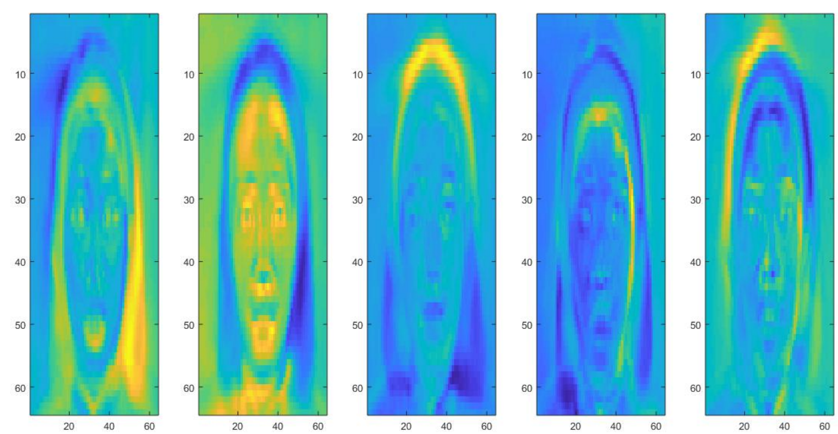

Figure 1: The result of the top 5 eigenfaces

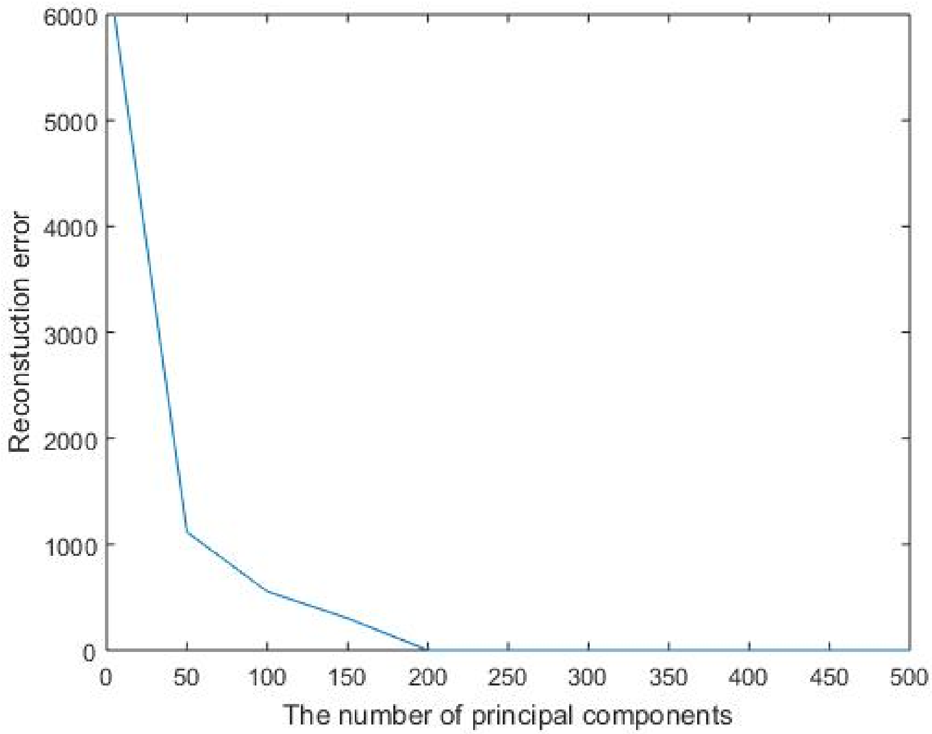

Figure 2: reconstruction error graph

Reconstruction error for 5 principal component: 5.979e+03
Reconstruction error for 50 principal component: 1.1145e+03
Reconstruction error for 200 principal component: 1.7966e-10
Reconstruction error for 500 principal component: 1.7966e-10