Module 1 – Making Sense of Unstructured Data

CASE STUDY ACTIVITY TUTORIAL

CASE STUDY 1 – PCA: Computing Eigenfaces

PROFESSIONAL EDUCATION
Digital Programs

CASE STUDY ACTIVITY TUTORIAL

CASE STUDY 1 – PCA: Computing Eigenfaces

Faculty Name: Stefanie Jegelka

In this document, we walk through some helpful tips to get you started with building your own application for classifying faces in photo images using Principle Component Analysis (PCA). In this tutorial, we provide examples and some pseudo-code for the following programming environment: **Matlab**. We cover the following:

Table of Contents

Image Dataset

You are encouraged to try this example using your own collection of images. You should save each image separately and they should all have the following characteristics:

1. The face should cover a large section of the image area.
2. As much as possible, the person in the image should be looking directly at the camera.

For reference, we provide the images of the instructors of this course. These images are all in jpg format and titled as "*1.jpg*", "*2.jpg*", ... "*14.jpg*". Ensure that they are placed in a directory called "*instructors*". This directory is assumed to be placed in the same folder where your Matlab script will be located.

Processing and Visualizing the Dataset

Given that the raw images are not all the same dimensions and in RGB format, we will need to pre-process them to all be of the same dimensions and in Grayscale color. The following lines of Matlab code can help us achieve this.

```
% number of images on the training set.
M = 14;

%read and show images(jpg);
```

```
% S will store all the images

S=[];

figure(1);
for i=1:M

    str = strcat('instructors/', int2str(i));
    str = strcat(str, '.jpg');
    eval('img=imread(str);');
    img = rgb2gray(img);
    img = imresize(img, [300,300]);

    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    if i==3
        title('Course Intructors','fontsize',14)
    end
    drawnow;

    % save the dimensions of the image (irow, icold)
    [irow, icol]=size(img);

    % creates a (N1*N2) x 1 matrix and add to S
    temp=reshape(img',irow*icol,1);

    %S will eventually be a (N1*N2) x M matrix.
    S=[S temp];

end
```

Computing Eigenfaces

In order to compute all Eigenfaces, we need to perform several processing steps. Please ensure you perform the following in order.

1. We first need to normalize the each image. Note that the normalized, square, gray images are saved in the same location as the Matlab script and will be reused several times. The following lines of code can be used to achieve this, and also display the normalized images:

```
%Normalize.
for i=1:size(S,2)
    temp=double(S(:,i));
    m=mean(temp);
    st=std(temp);
    S(:,i)=(temp-m)*ustd/st+um;
end

%save and show normalized images
figure(2);
for i=1:M
    str=strcat(int2str(i),'.jpg');
    img=reshape(S(:,i),icol,irow);
```

```
        img=img';
        eval('imwrite(img,str)');
        subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
        imshow(img)
        drawnow;
        if i==3
            title('Normalized Images','fontsize',18)
        end
    end
```

2. Compute a Mean (Average) face from all the normalized faces. The following lines of code will compute and show the average face:

```
%mean face;

%obtains the mean of each row of each image
m=mean(S,2);

%convert to unsigned 8-bit integer. Values range from 0 to 255
tmimg=uint8(m);

%takes the vector and creates a matrix
img=reshape(tmimg,icol,irow);

%matrix transpose
img=img';

figure(3);
imshow(img);
title('Mean Image','fontsize',18)

meanImg = img
```

3. (Optional) If you want to see each normalized image as a difference off the Mean face, you can use the following lines of code:

```
% show the difference from mean
figure(4);
for i=1:M
    str=strcat(int2str(i),'.jpg');
    img=reshape(S(:,i),icol,irow);
    img=img';
    img = img - meanImg;
    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    drawnow;
    if i==3
        title('Difference of Normalized Images from the
Mean','fontsize',18)
    end
end
```

4. Now, we can compute the Eigenfaces (Eigenvectors). The following code computes all the necessary components needed to compute the Eigenfaces and then finally saves (under the directory: "*eigenfaces*") and shows the Eigenfaces:

```
% Compute A matrix
dbx=[];
for i=1:M
    temp=double(S(:,i));
    dbx=[dbx temp];
end

A=dbx';

%Covariance matrix C=A'A, L=AA'
L=A*A';
% vv are the eigenvector for L
% dd are the eigenvalue for both L=dbx'*dbx and C=dbx*dbx';
[vv dd]=eig(L);

% Sort and eliminate those whose eigenvalue is zero
v=[];
d=[];
for i=1:size(vv,2)
    if(dd(i,i)>1e-4)
        v=[v vv(:,i)];
        d=[d dd(i,i)];
    end
 end

 %sort, will return an ascending sequence
 [B index]=sort(d);
 ind=zeros(size(index));
 dtemp=zeros(size(index));
 vtemp=zeros(size(v));
 len=length(index);
 for i=1:len
    dtemp(i)=B(len+1-i);
    ind(i)=len+1-index(i);
    vtemp(:,ind(i))=v(:,i);
 end
 d=dtemp;
 v=vtemp;

%Normalization of eigenvectors
 for i=1:size(v,2)
   kk=v(:,i);
   temp=sqrt(sum(kk.^2));
   v(:,i)=v(:,i)./temp;
end

%Eigenvectors of C matrix
u=[];
```

```
for i=1:size(v,2)
    temp=sqrt(d(i));
    u=[u (dbx*v(:,i))./temp];
end

%Normalization of eigenvectors of the C matrix
for i=1:size(u,2)
   kk=u(:,i);
   temp=sqrt(sum(kk.^2));
    u(:,i)=u(:,i)./temp;
end

% show eigenfaces;
EigenFaces = [];
figure(5);
for i=1:size(u,2)
    img=reshape(u(:,i),icol,irow);
    img=img';
    img=histeq(img,255);

    str = strcat('eigenimages/', int2str(i));
    str = strcat(str, '.jpg');
    eval('imwrite(img,str)');

    EigenFaces = [EigenFaces img];

    subplot(ceil(sqrt(M)),ceil(sqrt(M)),i)
    imshow(img)
    drawnow;
    if i==3
        title('Eigenfaces','fontsize',18)
    end
end
```

Classifying New Faces

In order for us to reconstruct new (unseen) based on the eigenfaces computed earlier, we will need to first compute the weights of each eigenface in the training set. Then, we will compute the weights obtained for the new image. Finally, we will compute the distance between the weight of the new image from the ones obtained for the training set. The least distance corresponds to the training set image we will classify the new image as. The following Matlab code can help with this classification:

```
% Find the weight of each face for each image in the training set.
% omega will store this information for the training set.
omega = [];
for h=1:size(dbx,2)
    WW=[];
    for i=1:size(u,2)
        t = u(:,i)';
        WeightOfImage = dot(t,dbx(:,h)');
        WW = [WW; WeightOfImage];
```

```
    end
    omega = [omega WW];
end

% InputImage is the new (unseen) image
% ensure that it is the same dimension image as the training set
% We assume this new image is titled 'new_image.jpg'
img=imread('new_image.jpg')
InputImage = rgb2gray(img);

figure(5)
subplot(1,2,1)
imshow(InputImage); colormap('gray');title('New image','fontsize',18)
InImage=reshape(double(InputImage)',irow*icol,1);
temp=InImage;
me=mean(temp);
st=std(temp);
temp=(temp-me)*ustd/st+um;
NormImage = temp;

p = [];
aa=size(u,2);
for i = 1:aa
    pare = dot(NormImage,u(:,i));
    p = [p; pare];
end

%m is the mean image, u is the eigenvector
ReshapedImage = m + u(:,1:aa)*p;
ReshapedImage = reshape(ReshapedImage,icol,irow);
ReshapedImage = ReshapedImage';

%show the reconstructed image.
subplot(1,2,2)
imagesc(ReshapedImage); colormap('gray');
title('Reconstructed image','fontsize',18)

% Compute the weights of the eigenfaces in the new image
InImWeight = [];
for i=1:size(u,2)
    t = u(:,i)';
    WeightOfInputImage = dot(t,Difference');
    InImWeight = [InImWeight; WeightOfInputImage];
end

% Find distance
e=[];
for i=1:size(omega,2)
    q = omega(:,i);
    DiffWeight = InImWeight-q;
    mag = norm(DiffWeight);
    e = [e mag];
```

```
end

kk = 1:size(e,2);
subplot(1,2,2)
stem(kk,e)
title('Eucledian distance of input image from each training
image','fontsize',14)
```

Reference:

Santiago Serrano, www.pages.drexel.edu/~sis26/Eigencode.html
Note: this url is no longer valid since the web portal does not exist any longer.