Overview of Topics

(1) SVD and Machine Learning

Example: Eigenfaces & facial recognition

Example: Proper orthogonal decomposition and physics (fluid flows...)

1.6 Eigenfaces Example

One of the most striking demonstrations of SVD/PCA is the so-called eigenfaces example. In this problem, PCA (i.e. SVD on mean-subtracted data) is applied to a large library of facial images to extract the most dominant correlations between images. The result of this decomposition is a set of *eigenfaces* that define a new coordinate system. Images may be represented in these coordinates by taking the dot product with each of the principal components. It will be shown in Chapter **??** that images of the same person tend to cluster in the eigenface space, making this a useful transformation for facial recognition and classification [51, 4]. The eigenface problem was first studied by Sirovich and Kirby in 1987 [49] and expanded on in [31]. Its application to automated facial recognition was presented by Turk and Pentland in 1991 [53].

Here, we demonstrate this algorithm using the Extended Yale Face Database B [16], consisting of cropped and aligned images [34] of 38 individuals (28 from the extended database, and 10 from the original database) under 9 poses and 64 lighting conditions[7]. Each image is 192 pixels tall and 168 pixels wide. Unlike the previous image example in Section 1.2.2, each of the facial images in our library have been reshaped into a large column vector with $192 * 168 = 32256$ elements. We use the first 36 people in the database (left panel of Fig. 1.16) as our training data for the eigenfaces example, and we hold back two people as a test set. An example of all 64 images of one specific person are shown in the right panel. These images are loaded and plotted using Codes 1.15 and 1.16.

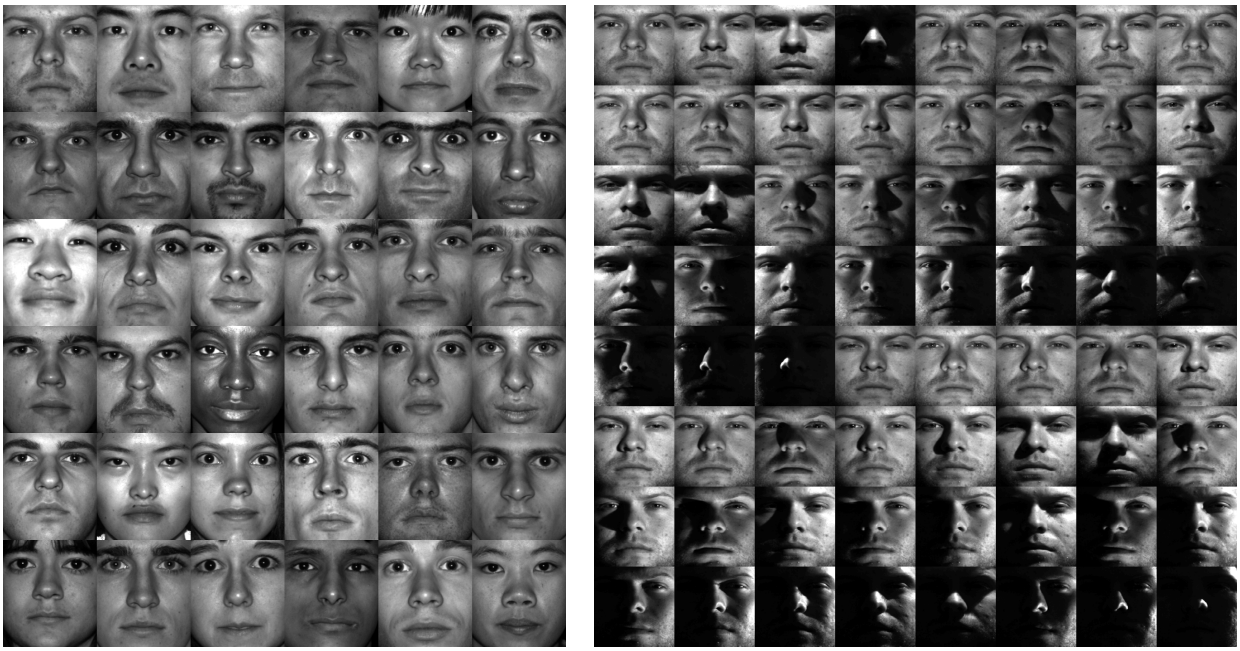

Figure 1.16: (left) A single image for each person in Yale database, and (right) all images for a specific person. Generated by Codes (1.15) & (1.16).

[7]The database can be downloaded at `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`.

Code 1.15: Plot an image for each person in Yale database (Fig. 1.16 (a))

```
clear all, close all, clc

load ../DATA/allFaces.mat

allPersons = zeros(n*6,m*6);
count = 1;
for i=1:6
    for j=1:6
        allPersons(1+(i-1)*n:i*n,1+(j-1)*m:j*m) ...
            = reshape(faces(:,1+sum(nfaces(1:count-1))),n,m);
        count = count + 1;
    end
end

figure(1), axes('position',[0  0  1  1]), axis off
imagesc(allPersons), colormap gray
```

Code 1.16: Plot each image for a specific person in Yale database (Fig. 1.16 (b))

```
for person = 1:length(nfaces)
    subset = faces(:,1+sum(nfaces(1:person-1)):sum(nfaces(1:person))
       );
    allFaces = zeros(n*8,m*8);

    count = 1;
    for i=1:8
        for j=1:8
            if(count<=nfaces(person))
                allFaces(1+(i-1)*n:i*n,1+(j-1)*m:j*m) ...
                    = reshape(subset(:,count),n,m);
                count = count + 1;
            end
        end
    end

    imagesc(allFaces), colormap gray
end
```

As mentioned before, each image is reshaped into a large column vector, and the average face is computed and subtracted from each column vector. The mean-subtracted image vectors are then stacked horizontally as columns in data matrix $\mathbf{X}$, as shown in Fig. 1.17. Thus, taking the SVD of the mean-subtracted matrix $\mathbf{X}$ results in the PCA. The columns of $\mathbf{U}$ are the eigenfaces, and they may be reshaped back into 192×168 images. This is illustrated in Code 1.17.

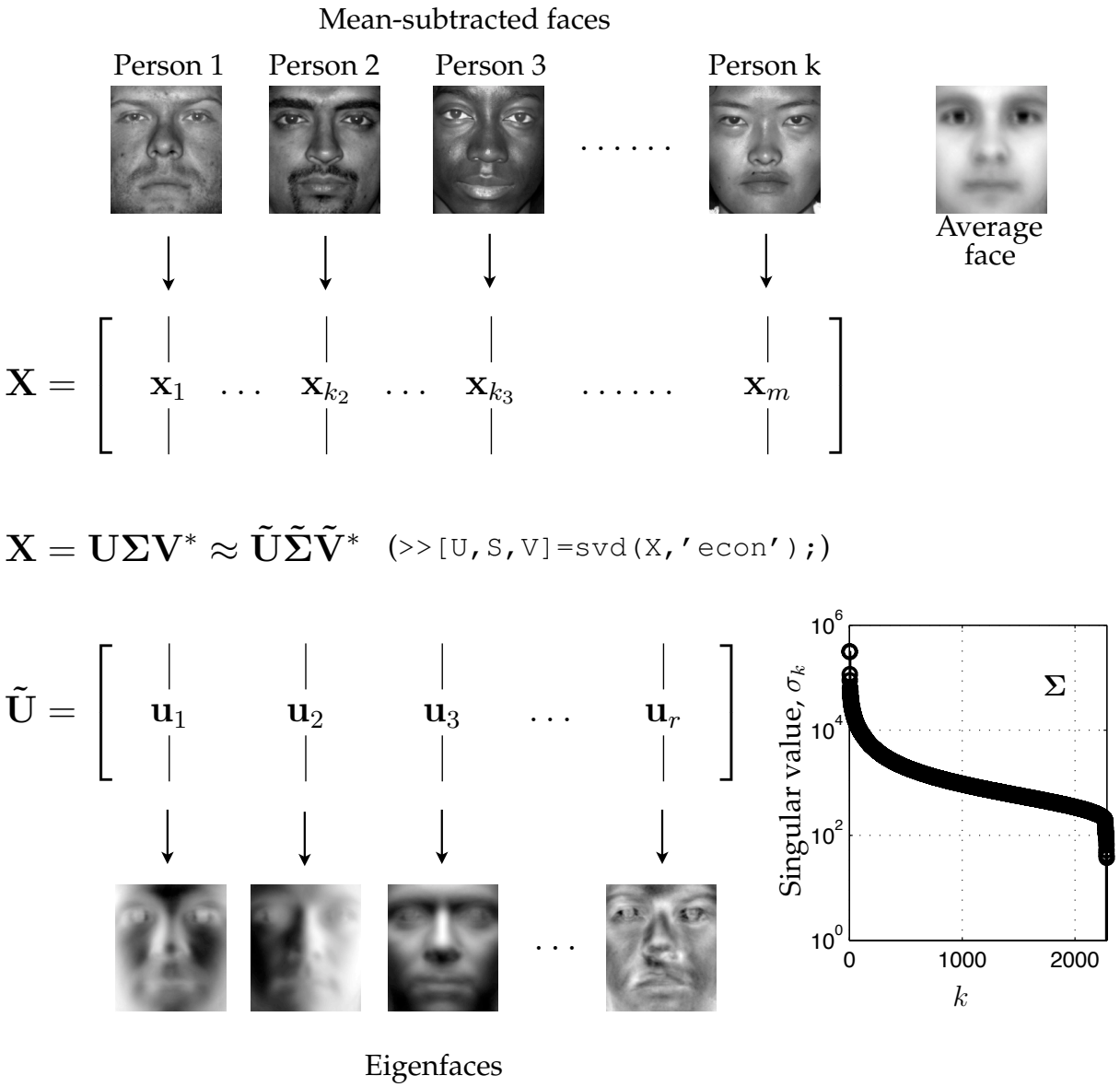

Figure 1.17: Schematic of procedure to obtain eigenfaces from library of facial images.

Code 1.17: Compute eigenfaces on mean-subtracted data.

```
% We use the first 36 people for training data
trainingFaces = faces(:,1:sum(nfaces(1:36)));
avgFace = mean(trainingFaces,2);  % size n*m by 1;

% compute eigenfaces on mean-subtracted training data
X = trainingFaces-avgFace*ones(1,size(trainingFaces,2));
[U,S,V] = svd(X,'econ');

figure, axes('position',[0  0  1  1]), axis off
```

```
imagesc(reshape(avgFace,n,m)), colormap gray

for i=1:50  % plot the first 50 eigenfaces
    pause(0.1);  % wait for 0.1 seconds
    imagesc(reshape(U(:,i),n,m)); colormap gray;
end
```

Using the eigenface library, ($\mathbf{U}$), obtained above, we now attempt to approximately represent an image that was not in the training data. At the beginning, we held back two individuals (the 37^{th} and 38^{th} people), and we now use one of their images as a test image, $\mathbf{x}_{\text{test}}$. We will see how well a rank-r SVD basis will approximate this image using the following projection:

$$\tilde{\mathbf{x}}_{\text{test}} = \tilde{\mathbf{U}}_r \tilde{\mathbf{U}}_r^* \mathbf{x}_{\text{test}}.$$

The eigenface approximation for various values of r is shown in Fig. 1.18, as computed using Code 1.18. The approximation is relatively poor for $r \leq 200$, although from $r = 400$ to $r = 1600$ it converges to a passable representation of the test image.

It is interesting to note that the eigenface space is not only useful for representing human faces, but may also be used to approximate a dog (Fig. 1.19) or a cappuccino (Fig. 1.20). This is possible because the 1600 eigenfaces span a large subspace of the 32256 dimensional image space corresponding to broad, smooth, non-localized spatial features, such as cheeks, forehead, mouths, etc.

Figure 1.18: Approximate representation of test image using eigenfaces basis of various order r. Test image is not in training set.

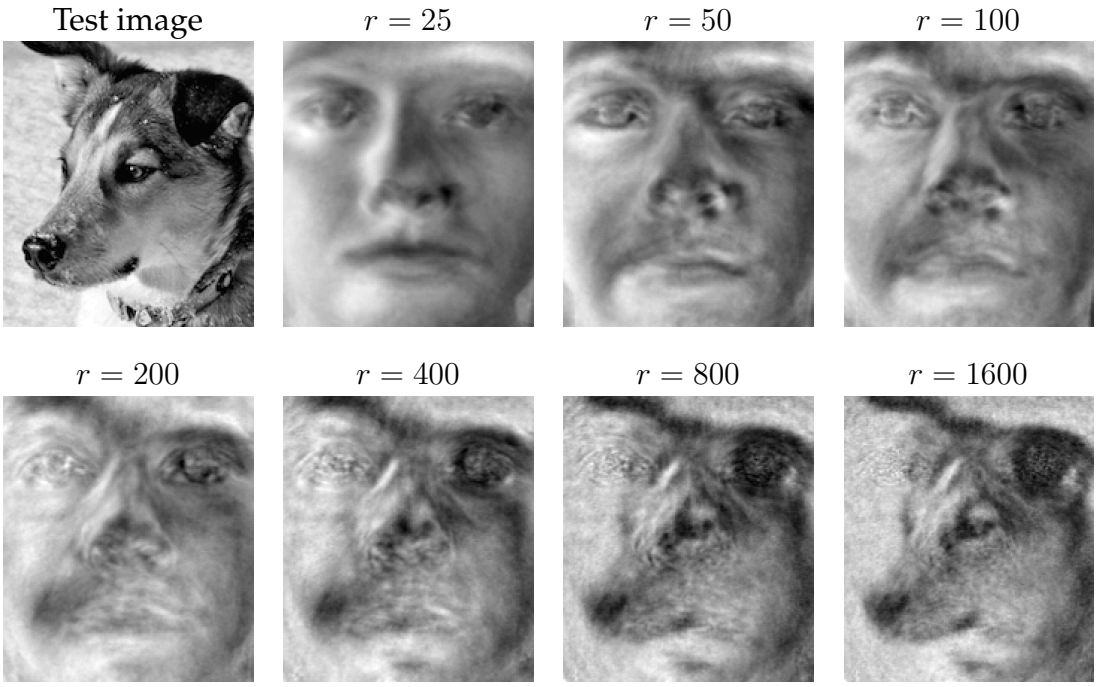

Figure 1.19: Approximate representation of an image of a dog using eigenfaces.

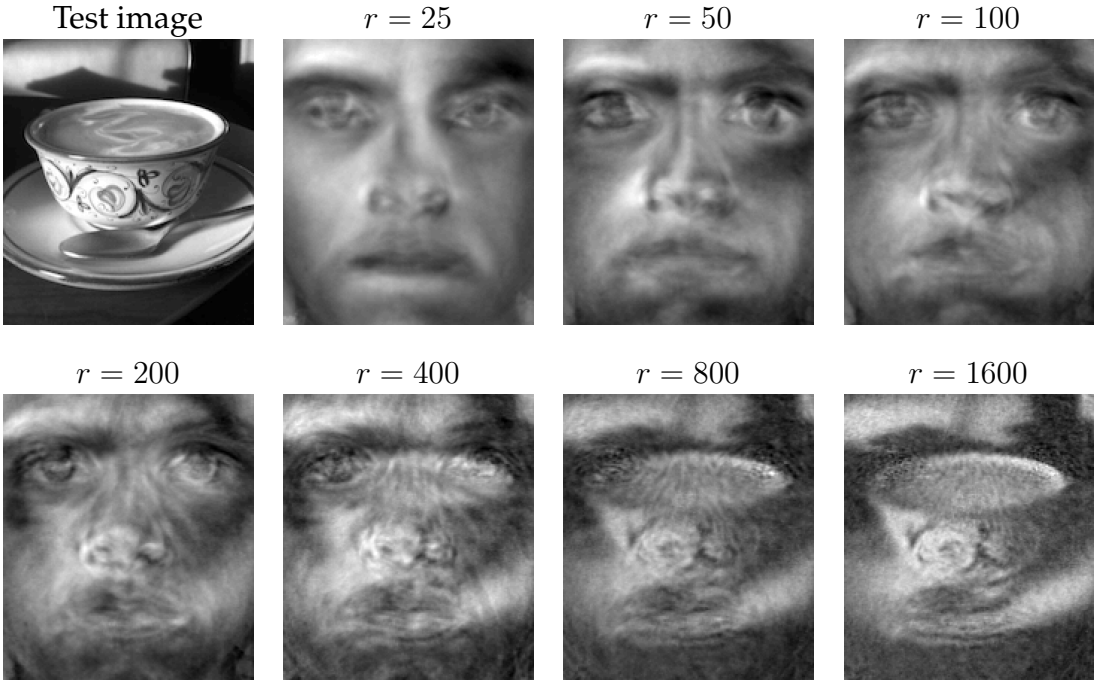

Figure 1.20: Approximate representation of a cappuccino using eigenfaces.

Code 1.18: Approximate test-image that was omitted from training data, using eigenfaces.

```
testFace = faces(:,1+sum(nfaces(1:36))); % first face of person 37
axes('position',[0  0  1  1]), axis off
imagesc(reshape(testFace,n,m)), colormap gray

testFaceMS = testFace - avgFace;
for r=25:25:2275
    reconFace = avgFace + (U(:,1:r)*(U(:,1:r)'*testFaceMS));
    imagesc(reshape(reconFace,n,m)), colormap gray
    title(['r=',num2str(r,'%d')]);
    pause(0.1)
end
```

We further investigate the use of the eigenfaces as a coordinate system, defining an eigenface space. By projecting an image $\mathbf{x}$ onto the first r PCA modes, we obtain a set of coordinates in this space: $\alpha = \tilde{\mathbf{U}}_r^* \mathbf{x}$. Some principal components may capture the most common features shared among all human faces, while other principal components will be more useful for distinguishing between individuals. Other principal components may capture differences in lighting angles. Figure 1.21 shows the coordinates of all 64 images of two individuals projected onto the 5^{th} and 6^{th} principal components, generated by Code 1.19. Images of the two individuals appear to be well-separated in these coordinates. This is the basis for image recognition and classification in Chapter **??**.

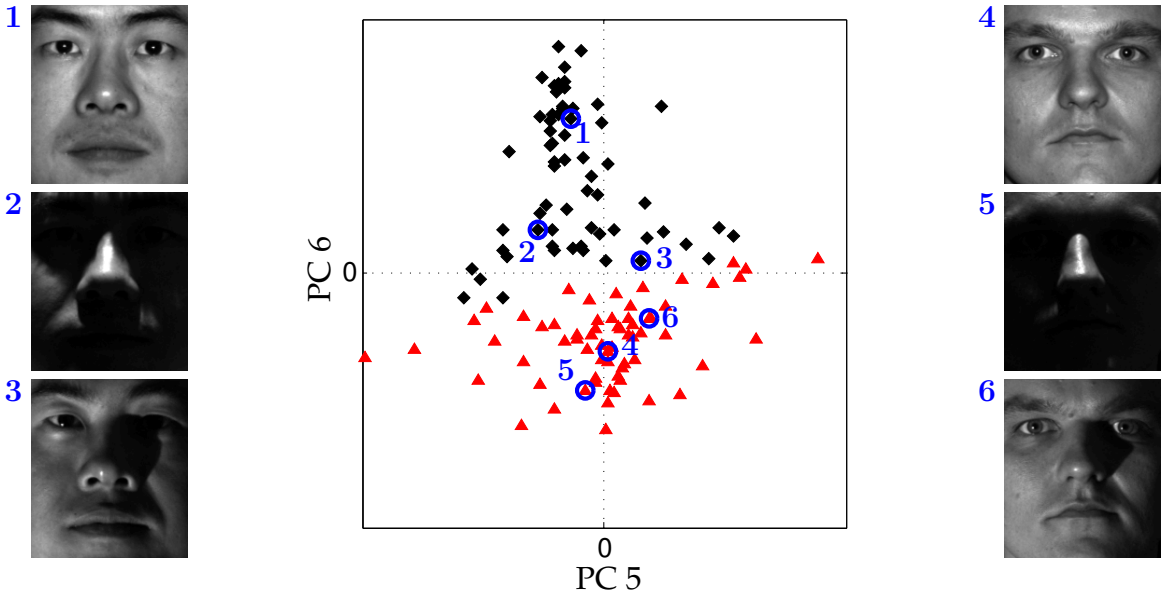

Figure 1.21: Projection of all images from two individuals onto the 5^{th} and 6^{th} PCA modes. Projected images of the first individual are indicated with black diamonds, and projected images of the second individual are indicated with red triangles. Three examples from each individual are circled in blue, and the corresponding image is shown.

Code 1.19: Project images for two specific people onto the 5^{th} and 6^{th} eigenfaces.

```
P1num = 2;  % person number 2
P2num = 7;  % person number 7
P1 = faces(:,1+sum(nfaces(1:P1num-1)):sum(nfaces(1:P1num)));
P2 = faces(:,1+sum(nfaces(1:P2num-1)):sum(nfaces(1:P2num)));
P1 = P1 - avgFace*ones(1,size(P1,2));
P2 = P2 - avgFace*ones(1,size(P2,2));

figure
subplot(1,2,1), imagesc(reshape(P1(:,1),n,m)); colormap gray, axis
   off
subplot(1,2,2), imagesc(reshape(P2(:,1),n,m)); colormap gray, axis
   off

% project onto PCA modes 5 and 6
PCAmodes = [5 6];
PCACoordsP1 = U(:,PCAmodes)'*P1;
PCACoordsP2 = U(:,PCAmodes)'*P2;

figure
plot(PCACoordsP1(1,:),PCACoordsP1(2,:),'kd','MarkerFaceColor','k')
axis([-4000 4000 -4000 4000]), hold on, grid on
plot(PCACoordsP2(1,:),PCACoordsP2(2,:),'r^','MarkerFaceColor','r')
set(gca,'XTick',[0],'YTick',[0]);
```