can be solved. Aside from requiring the det$\mathbf{A} \neq 0$, we also have a solvability condition on $\mathbf{b}$. Consider the adjoint problem

$$\mathbf{A}^\dagger \mathbf{y} = 0 \tag{2.4.31}$$

where $\mathbf{A}^\dagger = \mathbf{A}^{*T}$ is the adjoint which is the transpose and complex conjugate of the matrix $\mathbf{A}$.

The definition of the adjoint is such that

$$\mathbf{y} \cdot \mathbf{A}\mathbf{x} = \mathbf{A}^\dagger \mathbf{y} \cdot \mathbf{x}. \tag{2.4.32}$$

Since $\mathbf{A}\mathbf{x} = \mathbf{b}$, the left side of the equation reduces to $\mathbf{y} \cdot \mathbf{b}$ while the right side reduces to $\mathbf{0}$ since $\mathbf{A}^\dagger \mathbf{y} = \mathbf{0}$. This then gives the condition

$$\mathbf{y} \cdot \mathbf{b} = \mathbf{0} \tag{2.4.33}$$

which is known as the *Fredholm-alternative* theorem, or a *solvability* condition. In words, the equation states that in order for the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ to be solvable, the right-hand side forcing $\mathbf{b}$ must be orthogonal to the null space of the adjoint operator $\mathbf{A}^\dagger$.

2.5 Eigenvalues and Eigenvectors for Face Recognition

Normally, to motivate the concept of what eigenvalues and eigenvectors mean, examples would be given from some physical and/or engineering system. Often, however, the insight given into the physical system assumes that you actually know the workings of the physical system fairly well. Thus eigenvalues and eigenvectors often fail to impress students as they should. As a generic statement, it is difficult to refute the claim that if you know the eigenvectors and eigenvalues of a given system (or matrix), then you know everything there is to know about that system. Given then their importance, some measure of intuition will be developed about their meaning and use.

To motivate the importance of eigenvalues and eigenvectors, they will be considered in the context of the computer vision problem of human face recognition. The approach of using so-called *eigenfaces* for recognition was developed by Sirovich and Kirby [3, 4] and used by Turk and Pentland [5] in face classification. It is considered the first successful example of facial recognition technology. In what follows, some very basic ideas of eigenfaces will be given.

First, consider a set of face images. Figure 2.4 will be our so-called *training set* of images. This will be a highly limited example as we will consider only faces from four different people representing the political, entertainment and sports arenas, namely George Clooney, Barack Obama, Margaret Thatcher and Matt Damon. For each celebrity, five images are considered that are roughly cropped the same way. If you were a computer scientist, you would probably also make sure to subtract the background from the pictures, make sure all faces are looking directly at the camera without a head tilt, have photos with identical (or nomarlized) lighting, crop in a more systematic way, etc. However, this is only a rough guide on how the techniques work and

Figure 2.4: Five image samples of head shots of four different faces including George Clooney (top row), Barack Obama (second row), Margaret Thatcher (third row) and Matt Damon (bottom row). Each image was cropped, turned to gray-scale, and resized to 120 vertical and 80 horizontal pixels. All images are public domain, licensed under Creative Commons Attribution-Share Alike 2.0, or licensed under Creative Commons Attribution 3.0.

we will not concern ourselves with all of the extras that can make the eigenface methodology perform better.

To begin, all images are imported, turned into gray-scale, and resized to the same number of pixels in the vertical (120 pixels) and horizontal (80 pixels) directions. The following code takes an image called **pic.jpg**, converts it into gray-scale and turns the image format into double precision numbers for manipulation. Resizing is also accomplished.

```
A=imresize(double(rgb2gray(imread('pic','jpeg'))),[120 80]);
```

This preprocessing is done for each picture to create an image matrix **A**. A convenient way to view the image is with the **pcolor** command

```
pcolor(flipud(A)), shading interp, colormap(gray)
set(gca,'Xtick',[],'Ytick',[])
```

Note that the **flipud** command flips the image to be right-side-up for visualization. Further, the **imresize** command is part of the image processing toolbox.

To start understanding the face recognition process, we can begin by trying to understand the concept of an *average* face. In particular, if one takes the five images of a single person represented in Fig. 2.4 and averages over them, one would arrive at an average face for that person. Figure 2.5 demonstrates the average face associated with the five images of the four considered celebrities. The average faces are trivially computed by the following MATLAB code

```
Ave=(A1+A2+A3+A4+A5)/5;
```

where **A1** through **A5** are the five images of a particular celebrity. Note that although each celebrity image has a different background and is slightly misaligned from the others, the average still represents the basic look of the celebrity. In some sense, this is their *average* face. Interestingly enough, each picture can be thought of as the average plus corrections, or variance from the average. And such is the fundamental philosophy of the approach of eigenfaces: there exists an average face and all other faces are simply variances (perhaps small or large) from the average face. Indeed, it is conjectured that each person's variance from the average face is unique so that

Figure 2.5: Average faces generated from the five images of each celebrity of Fig. 2.4. Despite the varying backgrounds and image misalignments, the images retain the fundamental aspects of the individuals considered.

identification of the face can be made from them. To get a good average face for human beings, or even individuals or select groups of individuals, one would need very large data sets to improve the eigenface methodology.

What the average face actually emphasizes is the common features among a group of pictures, i.e. the pixels that are correlated. This gives a clue on how to pursue a face recognition algorithm: look for the correlations and common features of faces. Alternatively, two faces that are highly uncorrelated are most likely different. Correlations between data sets can be computed using the *covariance*. To do this, all the images in Fig. 2.4 are arranged into a matrix so that

$$\mathbf{B} = \begin{pmatrix} image1 \\ image2 \\ \vdots \\ imageN \end{pmatrix} \tag{2.5.1}$$

where each image has been reshaped into a vector using the command

```
a=reshape(A,1,120*80)
```

The new matrix **B** now has individual images that comprise each row. In our case, this will be a matrix with 20 rows with 120×80 columns.

To compute a correlation matrix, it only remains to multiply each row vector by each other row vector. Specifically, the inner product of any two rows gives the correlation between those two images. One can also normalize this if desired, but we will not do so here. Further, most eigenface methodologies subtract the average face of the entire data set so that everything is about the deviation from this average face. Again, this is not necessary to demonstrate the method. To correlate all the pixels and images, i.e. to multiply all rows by all other rows, we can simply compute the correlation matrix

$$C = B^T B \tag{2.5.2}$$

where the superscript T denotes the transpose of the data matrix. This results in a very large correlation matrix (9600×9600).

Our objective is to then compute the eigenvalues and eigenvectors of this matrix in order to gain some insight into the face recognition algorithm. To compute the eigenvalues and eigenvectors, simply apply the code from the last section

```
C=B' * B;
[V,D]=eigs(C,20,'lm');
```

where the first 20 eigenvalues and eigenvectors are generated with the largest magnitude. The matrix **V** contains the first eigenvectors as the columns and the vector **D** contains the first

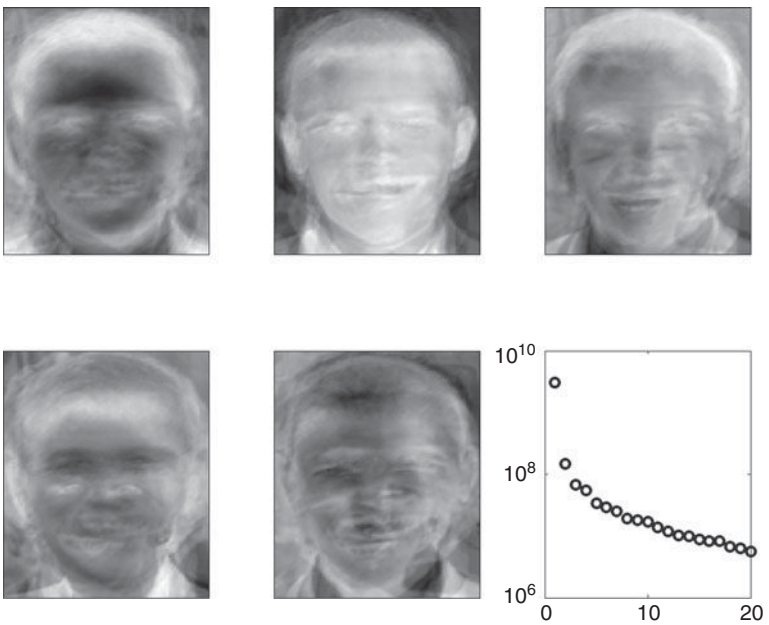

Figure 2.6: The first five eigenvectors (or eigenface components) of the correlation matrix **C** arranged from top left to bottom right. In the bottom right panel the first 20 eigenvalues are shown. Note the dominance of the first mode on the log scale which suggests it captures the average face in the data matrix. The additional eigenvectors are added in a weighted fashion to the average mode in order to produce the individual faces (or variances) of each image.

20 eigenvalues as its diagonals. Figure 2.6 shows the first five eigenvectors (reshaped back to matrix form for viewing) of our data matrix. Additionally, the first 20 eigenvalues are demonstrated showing the decrease in their value. The magnitude of the eigenvalues is directly related to how important its eigenvector is in composing the images. In the log plot of Fig. 2.6, the first mode dominates the *energy* content of all the images. This is effectively the average face within the 25 selected images. The other modes are shown as a function of decreasing importance. The idea is that one can reconstruct any of the faces considered as a weighted sum of the eigenvectors shown. All that remains is a demonstration on how the weighting of the eigenvectors is achieved. Given the dominance of the first few eigenvectors and their importance in the weighting, they are often called *principal components*, or a principal component analysis (PCA). As the name implies, one can think of them as the principal quantities (or most important quantities) of interest in the face recognition algorithm. A great deal more will be discussed concerning PCA in the third part of this book on data analysis. What is important here is that the eigenvalue/eigenvector decomposition of the matrix highlights the most essential features of the data matrix and also ranks, or prioritizes, through the eigenvalues the importance of each eigenvector.

To represent an individual in terms of the eigenvectors of the full data matrix **C**, it remains to understand the weighting of each image on the eigenvectors. This can be done by simply

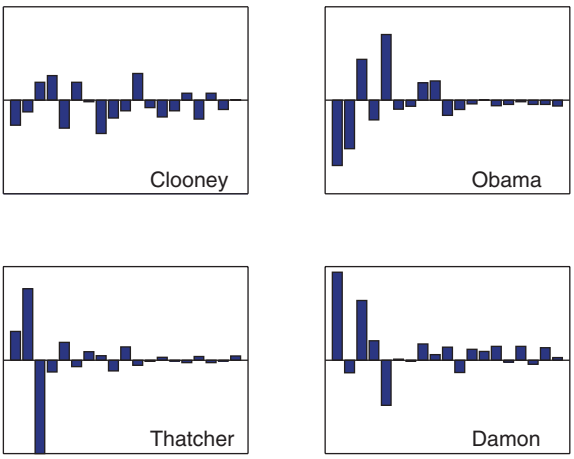

Figure 2.7: Projection of the average faces in Fig. 2.5 onto the eigenvector space computed from the full correlation matrix **C**. Note that each celebrity has a different, and hopefully unique, set of coefficients that can be used for face recognition.

computing the inner product of each image on each eigenvector. This can be done very simply with the MATLAB command

```
proj_a=a*V;
```

where it should be recalled that **a** is the reshaped picture in vector form of a specific image. Multiplying by the eigenvector matrix **V** gives a vector of inner products of the image onto each eigenvector. This operation can be performed for the average image (see Fig. 2.5) of the four celebrities considered in Fig. 2.4. This gives a *unique* representation of each celebrity on the eigenvectors. Figure 2.7 shows the projection coefficients of the four average faces of Fig. 2.5. Note that each celebrity has a different set of coefficients in this representation. It is these differences that can be used as the basis for face recognition.

To demonstrate how the eigenvectors can be used in practice, we consider reconstruction of a *new* image using the eigenvectors formed from our *training set*. Figure 2.8 shows three new images that are not part of the training set. In the first, a new picture of Margaret Thatcher (Fig. 2.8 top row) is introduced and projected onto the first 20 eigenvalues. The weighting coefficients are computed and a 20 mode representation is then constructed in the third panel of the top row. This shows that some semblance of Margaret Thatcher can indeed be reconstructed from our training set which included five different pictures of Margaret Thatcher. In the last panel of the top row, the difference between the projection vector of the new image and the projection vector of the original five Margaret Thatcher images is given. Thus the following quantity is computed

$$E_j = \frac{\|c_j - c_{\text{new}}\|}{\|c_j\|} \tag{2.5.3}$$

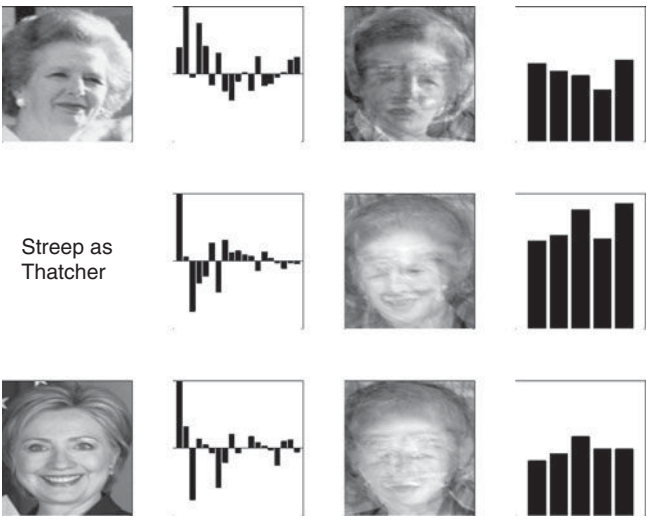

Figure 2.8: Projection of three new images (left panels) onto the eigenvector space created from Fig. 2.4. The coefficients of the projection are shown in the second column of data panels while the reconstruction using 20 eigenvectors is shown in the third column of data. The fourth column shows the distance in the coefficient measure (2.5.3) for each new picture against the original five Margaret Thatcher images. The new Margaret Thatcher image is able to produce a reasonable reconstruction while Meryl Streep (as Thatcher) and Hillary Clinton both perform poorly under reconstruction. All images are public domain, licensed under Creative Commons Attribution-Share Alike 2.0, or licensed under Creative Commons Attribution 3.0.

where the double bar denotes the L^2 norm, c_j is the projection of the jth Margaret Thatcher image in the original set of Fig. 2.4 and c_{new} is the new image presented. In MATLAB, this can be accomplished with

```
E=norm(proj_a-proj_new)/norm(proj_a);
```

This gives some measure of the difference between the new images and the original images already used in the correlation matrices. The conjecture is that if the image is similar to those in the training set, then this error, or difference, should be small. In the second row of Fig. 2.8, the results are reproduced with a new picture (not shown) of Meryl Streep who plays Margaret Thatcher in the movie *Iron Lady* (2012). We can compare her Margaret Thatcher look to the real Margaret Thatcher to conclude that with a 20 mode reconstruction (second row, third panel), she does not look much like Margaret Thatcher from the perspective of the training set used. The last panel in the second row shows the distance measure from the new picture to the five originals in terms of the coefficients of the first 20 eigenvalues. Note that if a totally different person is projected onto the data set (Hillary Clinton in the last row of Fig. 2.8), then no recognizable face is produced in the 20 mode reconstruction. Interestingly, however, Hillary Clinton does appear to be closer in the coefficient space to Margaret Thatcher than Meryl Streep.

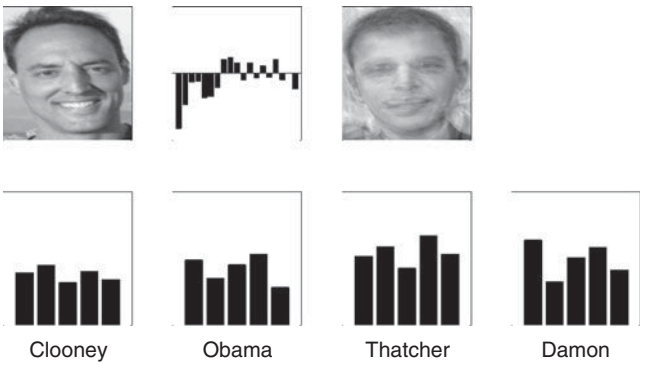

Figure 2.9: Projection of the author onto the eigenvector space created from Fig. 2.4. The coefficients of the projection are shown in the second panel of the top row while the reconstruction using 20 eigenvectors is shown in the third column. It seems like there is some resemblance to Obama and Damon. The bottom row shows the distance in the coefficient measure (2.5.3) for the author's picture against the original 25 images, showing that the author looks most like Obama than the others.

Finally, the author wishes to see how he might project onto this data set. This projection is shown in Fig. 2.9. In the top row, the original picture is shown along with the projection onto the coefficients of the eigenvectors and the reconstruction using 20 eigenvectors. What seems to be the case is that the author kind of looks like a cross between Barack Obama and Matt Damon. In fact, when comparing in the eigenvector coefficient space against all the images of Fig. 2.4, the distance calculation (2.5.3) shows the author to be quite similar to Obama/Damon in a couple of his pictures. The author was hoping to look perhaps more like George Clooney.

As a final note, the technique used in creating eigenfaces and using them for recognition is also used outside of facial recognition. This technique is also used for handwriting analysis, lip reading, voice recognition, sign language/hand gestures interpretation and medical imaging analysis. Therefore, some do not use the term eigenface, but prefer to use *eigenimage*.

2.6 Nonlinear Systems

The preceding sections deal with a variety of techniques to solve linear systems of equations. Provided the matrix $\mathbf{A}$ is nonsingular, there is one unique solution to be found. Often, however, it is necessary to consider a *nonlinear* system of equations. This presents a significant challenge since a nonlinear system may have no solutions, one solution, five solutions, or an infinite number of solutions. Unfortunately, there is no general theorem concerning nonlinear systems which can narrow down the possibilities. Thus even if we are able to find a solution to a nonlinear system, it may be simply one of many. Furthermore, it may not be a solution we are particularly interested in.

To help illustrate the concept of nonlinear systems, we can once again consider the Newton–Raphson method of Section 1.3. In this method, the roots of a function $f(x)$ are determined by the Newton–Raphson iteration method. As a simple example, Fig. 2.10 illustrates a simple cubic