Course: DD2427 - Exercise Set 7

In this set of exercises you will compute a two different set of eigenfaces from two different datasets. You will then use one set to reconstruct a frontal view of your own face.

Exercise 1: *Eigenfaces - Load training data*

To compute a set of eigenfaces training images are required. Fortunately, there are lots of images of faces. To start with, we will use the Bush images that you have previously used.

So your first task is to load these images and grab the centre patch which corresponds to the face. The prototype of the function will be

```
function [X, w, h] = LoadData(DirName, p)
```

In this function for each image you read in from `DirName` you should:

1. Convert it, if necessary, to a grayscale image. For this you can use the function `rgb2gray`.
2. Grab the center patch which is `p` percent of the image's area. (For reference - to create the George Bush basis given an image of size `w x h` I took the face patch to be a central patch which had height `h * .5` and width `w * .33`. While for the other face database I took the whole image as the face patch.)
3. Next normalize this pixel data to introduce some invariance to illumination effects. To do this compute the mean pixel intensity value, μ, and its standard deviation, σ, and then normalize:

   $$I(x, y) = \frac{I(x, y) - \mu}{\sigma}$$

   After this the mean of the pixel intensities will have mean 0 and standard deviation 1. Note that the *Matlab* functions `mean` and `std` can be used to compute the mean and standard deviation of a vector of numbers.
4. Then you should store this normalized pixel data as a column of `X`.

This function will return an array, `X`, of size `d`$\times$`n` where `d` is the number of pixels in each image patch you extract and `n` is the number of images loaded.

You should probably keep a record of the original dimensions of the images you used to create X. These dimensions will be useful when displaying computed eigenfaces and when using the eigenface basis to reconstruct novel image patches.

Exercise 2: *Eigenfaces - Computation*

You now have a function that loads the pixel data required to compute the eigenfaces. The next step is to compute the eigenfaces. Before you do this take note of the following. As we said `X` is an array of size `d`×`n`. The covariance of the data contained in `X` can be computed as

$$\texttt{C = } \frac{1}{n} \texttt{* Xc * Xc';} \tag{1}$$

`Xc` is the same as `X` except that the mean vector of the training vectors has been subtracted from each column. `C` is a matrix of size `d`×`d`. The eigenvectors of `C` correspond to the eigenfaces. However, for the Bush example `d` is a very large number and if you try to use the *Matlab* command `eig` to compute the eigenvectors of this `d`×`d` matrix, you'll probably hang your machine. But luckily there is a way around this problem!

So what should you do if `d` is much greater that `n`? First compute the matrix:

$$\texttt{C1 = } \frac{1}{n} \texttt{* Xc' * Xc;} \tag{2}$$

This has size `n`×`n`. If `n` is of a reasonable size then you can compute the eigenvectors of `C1`. Now if `v` is an eigenvector of `C1` with corresponding eigenvalue `lambda` then

$$\texttt{v1 = Xc * v;} \tag{3}$$

is an eigenvector of `C` with corresponding eigenvalue `lambda`. **As part of this assignment show that this is indeed the case**.

Now you are ready to write the function to compute the eignfaces. The function will have the form

```
function [mu, W, D] = ComputePCABasis(X)
```

The function will perform the following

1. Compute the mean vector, `mu` from the columns of `X`. You should get a result that looks something like the first image in figure 1.

2. Centre the data by taking `mu` from each column of `X` to obtain `Xc`.

3. Check if `d>n`. If yes then you need to compute `C1` as defined in equation 2. Otherwise compute `C` as in equation 1.

4. Use the *Matlab* function `eig` to compute the eigenvalues of `C1` or `C`. If you compute the eignevectors of `C1` then you need to transform these eigenvectors as in equation 3. You now have your eigenfaces.

5. Ensure that each of the eignfaces has norm 1.

The output of the function is:

- `mu` the mean vector of the training data (size `d`×`1`),
- an array `W` of size `d`×`n`. Each column of `W` is an eigenface,
- a vector `D` of size `d`×`1` whose values are the eignvalues corresponding to the eigenfaces in `W`. These pop out from the eigen-decomopostion. Note that the function `diag` extracts the main diagonal from a matrix.

Important note please check the order in which the *Matlab* function `eig` outputs the eigenvectors. For me they were sorted according to the increasing values of the eigenvalues. This means the end columns of `W` are the interesting eignfaces.

Once you have written the function, run it on the Bush dataset. Remember you can turn an eigenface vector `v` into a 2d array via

```
im_v = reshape(v, [h, w]);
```

and then display it as normal. Figure 1 displays the top 19 eigenfaces I computed from this dataset. Check them against those that you have computed.

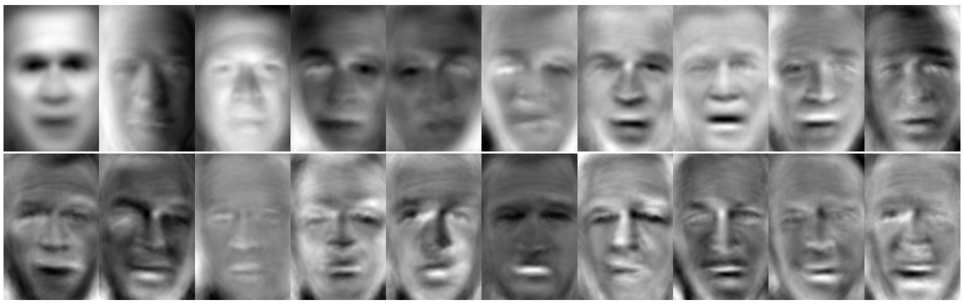
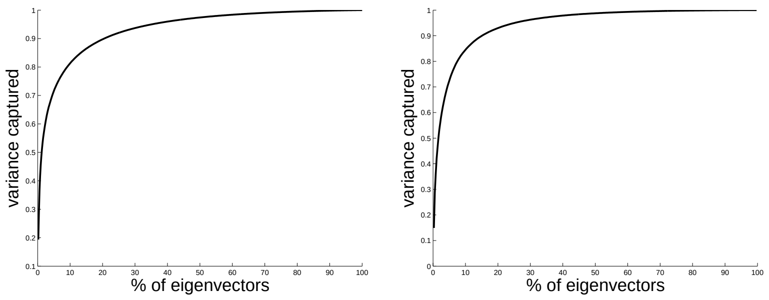


Figure 1: **The mean face and the top 19 eigenfaces from the Bush dataset.** The top left hand corner is the mean face. While the the rest of the images show the eigenfaces with the largest eigenvalues shown in descreasing order from left to right.

You can also analyze the eigenvalues associated with the eigenfaces. It is also interesting to plot the cumulative values of the eigenvalues `D` (divided by their total sum). The *Matlab* function `cumsum` allows you to do this. If you make this plot for the Bush dataset you should get something as in figure 2(a). Note I had to reverse the order of elements of `D` to make this plot so that the largest values were at the beginning not the end. **How many eignfaces are required to reconstruct 90% of the variation in the Bush dataset for the eigenface basis you constructed?**



(a) **Bush dataset** (b) **Small face dataset**

Figure 2: **Cumulative influence of the eigenvectors**. Thes graphs are computed by computing the cumulative sum of the eigenvalues associated with each eigenface when they are sorted in descending order. This sum is a measure of the percentage of variation in the training data captured by the eigenvectors included in the sum.

Exercise 3: *Face reconstruction using eigenfaces*

It was interesting to see what the main variations were with respect to photographs of George Bush's fronal face. However, using training examples from one person is not sufficient to build a basis that can be used to reconstruct images from a novel person. To be able to do this we will use another dataset and this can be downloaded from the course website. In this dataset the images cover a smaller part of the face, are very well aligned and are also much smaller (`19`×`19`).

Run the code you have just written to compute the mean face, eigenfaces, etc of this dataset. You may have to adapt your code slightly so that it grabs the whole image as opposed to the centre patch. Note that in this case generally the size of `C` will be smaller than `C1`. Thus you can perform eigendecomposition directly on `C`. Also you don't have to load all the images (use at least 3,000 to get nice results), though you can if you want but it may slow things down.

For reference, figure 3 displays the mean face and the top 19 eigenfaces I computed from using all the training data.

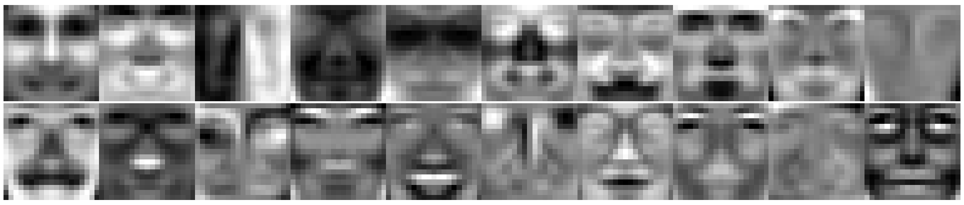


Figure 3: **The mean face and the top 19 eigenfaces from the second dataset.** The top left hand corner is the mean face. While the the rest of the images show the eigenfaces in descreasing order wrt to eigenvalue from left to right.

Now that you have a set of nice eigenfaces, the next task is to investigate if it can be used to reconstruct a novel face. Load the image `Student4.jpg` from the set of images associated with Exercise Set 4. Convert it to a greyscale image and then grab the patch

```
im = im(183:248, 297:362);
```

This is a patch that corresponds to one of the faces in the image. You will now write the function

```
function rim = ReconstructFace(im, mu, W, w, h, N)
```

that finds the best reconstruction of the image patch `im` using the top `N` eigenfaces.

1. Resize the image patch `im` to have size `[h,w]`. Use the *Matlab* function `imresize`. Then turn it into a column vector.
2. Normalize the patch so that its intensity values have mean 0 and standard deviation 1. Keep a record of the original mean value and standard deviation as these will be re-used at the end.
3. Project the normalized image patch minus `mu` onto each of the top `N` eigenfaces, i.e. the last `N` columns of `W`. Keep a record of these coefficients.
4. Approximate the original patch as a weighted sum of the top `n` eigenfaces (using the coefficients you just computed) plus the mean face.
5. Reverse the normalization you computed at the beginning.

6. Convert your reconstruction into a 2d array of size `[h, w]` using `reshape`. Then use `imresize` to resize it back to the size of the original patch. This is the output `rim` of your function.

Call this function with `N=1, 10, 20, 50, 100, 150, 200, 250` and see how the reconstruction improves. For reference, my results are shown in figure 4. You should note that this method only works well if the image patch we are trying to reconstruct is well aligned with the eigenfaces.

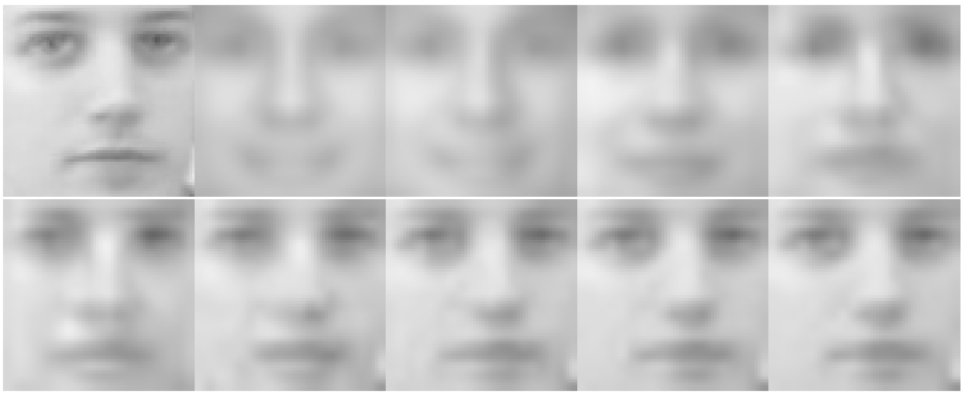


Figure 4: **Accuracy of the reconstruction increases with the number of eigenfaces used.** The top left hand corner shows the face we are trying to reconstruct. The rest of the images show the reconstruction when using the top 1, 2, 10, 20, 50, 10, 150, 200 and 250 eigenfaces.

For the lecture: 19th April

Bring the following:

- *a small proof of the question in Exercise 2 concerning the relationship between the eigenvalues of `C1` and `C`.*
- *a print out of the functions you wrote for this exercise `LoadData`, `ComputePCABasis`, `Reconstruct`.*
- *a print out of the top 10 eigenfaces computed from the first and second database used, the plot of the cumulative sum of the eigenvalues and the reconstruction of the `Student4.jpg` using 10, 50, 100, and 200 eigenfaces.*