Cogs 118B HW4 Python Instruction

`hw4data.mat` is a set of 100, 2-D data vectors arranged as columns in the matrix.

For B thru D, run `hw4data = np.loadtxt('hw4data.mat')` to load the data.

B)

1. Using the `matplotlib.pyplot.scatter` command with `'.'` marker, make a scatterplot of the samples. (and save as `hw4B.jpg` and put it in the document). (A scatter plot is just a plot showing each datapoint as a point in the 2-D space).
2. Find the sample mean using `numpy.mean`. Transform the returned 1D `numpy` array to a 2D `numpy` column vector. Note the mean should have shape `(2, 1)`.
3. Find the sample covariance matrix (and write it down in the document) using the formula $\frac{1}{n}ZZ^T$ where Z is the matrix of mean subtracted data points. You may **not** use built-in function for covariance. Check that your covariance matrix has the correct dimensionality (2×2).

(C)

1. Using the `eigvals, V = np.linalg.eig(covmatrix)` command, find the eigenvectors and eigenvalues of the sample covariance matrix.
2. What is the eigenvector corresponding to the largest eigenvalue?
3. What is the eigenvector corresponding to the smallest eigenvalue?
4. Use the `eigsort` function that we provide to sort the eigenvectors and eigenvalues in order of largest eigenvalue to smallest eigenvalue. If the matrix V_{old} is the eigenvectors and the array `eigvals` stores the eigenvalues, the syntax for `eigsort` is: $V_{sort}, D_{sort} = \text{eigsort}(V_{old}, \texttt{eigvals})$
5. How is the matrix of sorted eigenvectors related to the matrix of unsorted eigenvectors? (Look at them to confirm)

(D)

The PCA transformation of a point consists of subtracting the mean and multiplying by the transpose of the matrix of eigenvectors of the covariance matrix. $c = V' * (x - mean)$. This is just re-representing the point in a new data-aligned coordinate system.

1. Consider the point $\begin{bmatrix} -35 \\ 40 \end{bmatrix}$ in the original coordinate system. What are its coordinates in the new coordinate system found by PCA (i.e., what are the principal component coordinates of the point)? (Write in the document)

2. Transform all 100 sample points with the PCA transformation (you can do this with one big matrix multiplication ($C = V' * Z$ where C is a matrix with columns containing the principal component coordinates of the points, Z is a matrix with columns containing the mean subtracted data points, and V is the matrix of sorted eigenvectors) or in a loop using the equation used above). Using the `matplotlib.pyplot.scatter` command with `'.'` marker, make a scatterplot of the corresponding points in the new coordinate system. (Save as `hw4D.jpg` and include in the document)

3. How does `hw4D.jpg` compare to `hw4B.jpg`? (What kind of transformation takes you from `hw4B.jpg` to `hw4D.jpg`)? (write in the document)

For E thru I, run

```
hw4bdata = sio.loadmat('hw4bdata.mat')
faces = hw4bdata['faces']
dog = hw4bdata['dog']
x = hw4bdata['x']
```

to load the data.

The matrix `faces` contains 48 face images. These are images of 16 people in 3 different lighting conditions. Each column of `faces` is a 60x60 face image. These are the same faces that were used in Turk and Pentland's original paper on eigenfaces and the ones used in the demo of `faceexample09.m` (but different than the ones in the lecture slides). Feel free to look at the `faceexample09.m` code posted on the website for help with the various steps below.

!!! IMPORTANT: We use 1-based indexing in this writeup. So for example, when you are telling to view the 5th column, you might want to look at 0-based index 4.

(E)

1. Use the `viewcolumn` command to view the 5th (index 4) face image, which is stored in the 5th column of the `faces` matrix. (You don't need to print this face image out or put it in the document –it's just for you to see.)
2. Compute the mean face (use `numpy.mean`). Transform the returned 1D `numpy` array to a 2D `numpy` column vector. Then use `viewcolumn` to see it.
3. Subtract the mean from all of the data (using the command `numpy.matlib.repmat`), and call the matrix of mean-subtracted data `A`.
4. Use the eig command to compute the eigenvectors and eigenvalues of $A^T A$. (Note: we are not computing the eigenvectors and eigenvalues of AA^T directly because that is a very big matrix. See discussion of the transpose trick from class notes (related to question 6 from HW3).) Note that we have also dropped the $1/n$ term from the covariance matrix equation as it does not affect the eigenvectors (it just changes the eigenvalues, which are not used, by a factor of n).
5. Use `eigsort` to sort the eigenvectors and eigenvalues in order of largest to smallest eigenvalue. Call the matrix of sorted eigenvectors `V`.
6. Use `A` and `V` to calculate `U`, the matrix of eigenfaces (the eigenvectors of AA^T).

7. Use `normc` that we provide to normalize the columns of `U` so they all have length 1. Assign the output of normc to `U`.

(F)

For this exercise, just tell us the Python commands that you used (Put in the document).

Find the principal component coordinates of the 5th (index 4) face image in the data set. (This is just like what you did for question (D) 1. above). This is the re-representation of the zero-meaned fifth face in the new data-aligned coordinate space. Call the vector of principal component coordinates `c`.

(G)

1. Display the 3rd (index 2) eigenface. (use the `viewcolumn` command) (Save as `hw4G1.jpg` and put in the document)
2. Reconstruct the 5th face using all 48 principal components. (Remember that $\hat{z} = U_{reduced} * c_{reduced} + \texttt{meanface}$) where $U_{reduced}$ has as columns of the principal component directions that you wish to use and $c_{reduced}$ has the corresponding principal component coordinates). (For all reconstructions, remember to add back the mean face!) (Save as `hw4G2.jpg` and put in the document)
3. Reconstruct the 5th face using only the first 10 principal components. (Save as `hw4G3.jpg` and put in the document)

(H)

Let's explore what happens when a non-face image is projected into face space.

1. The vector `dog` contains a 60x60 picture of a dog. Display the picture of the dog. (Type the Python command into `HW4.txt`. You don't need to print this face image out or put in the document -it's just for you.)
2. Using the eigenfaces you found (Do NOT recompute the eigenfaces), find the principal component coordinates of the dog image (Type the Python command into the document). $c = U' * (\texttt{dog} - \texttt{meanface})$. This gives the coordinates of the dog image in the face-aligned space.
3. Reconstruct the dog picture using all 48 principal components, and display the image (Save as `hw4H.jpg` and put in the document). (This is just like (G2) except you use the principal component coordinates computed from the dog in H.2 above). This is the best reconstruction of the dog image in the face space. (Remember to add back in the meanface)
4. Does the reconstructed dog look like the original picture? Explain why the reconstruction looks the way it does (Type into the document).

(I)

Columns 1–3 of `faces` contain the three images of person 1, columns 4–6 contain the three images of person 2, and so on, all the way up to columns 46–48, which contain the three images of person 16. You will explore how well the face space generalizes to new people who weren't in the data set.

1. Choose one of the 16 people (3 of the 48 images) to exclude, and calculate the eigenfaces using only the remaining 15 people (45 images) (Type which three images you excluded in the document). You can leave faces 4-6 out by setting `newfaces = np.delete(faces,np.s_[3:6],axis=1)`
2. Choose one of the images of the excluded 16th person, and display the image (Save as `hw4I2.jpg` by using `plt.savefig`. Put the image in the document.)
3. Use the eigenfaces you found in part (1) to calculate the principal components of the image you chose in part (2). Reconstruct the image using all 45 of its principal components, and display (Save as `hw4I3.jpg` and put in the document)
4. Based on parts (1)–(3), how well do you think the face space represents faces that were not in the original data set?