This assignment is meant to exercise your probability and programming abilities. For both parts, you will need to produce a report which you will submit online. In addition, for the programming portion you will need to submit your code in a separate ZIP file. Both parts are due **Wednesday, April 5 at 11:59pm**. Late submissions will not be accepted. Submissions will be handled through Moodle.

# Mixture Models

Consider the Gaussian Mixture Model which assumes the data has been generated from the distribution

$$p(y|\theta) = \sum_{j=1}^{K} \pi_i \mathcal{N}(y|\mu_i, \Sigma_i)$$

where $\theta = \{(\pi_i, \mu_i, \Sigma_i)\}_{j=1}^{K}$ are the parameters of the mixture model.

1. In class we discussed fitting this model using the Expectation-Maximization (EM) algorithm. However, it's also possible to fit this model using gradient-based optimization. This can be very useful for GMMs when there are a large number of data points and can be used to fit mixtures of non-Gaussian distributions. However to do this the gradients of the objective function needs to be computed. In this question you'll derive the needed gradients and find a way to handle the constraints on the parameters. For simplicity, use the Maximum Likelihood objective

$$E(\theta) = -\sum_{i=1}^{N} \log p(y^{(i)}|\theta)$$

   (a) Derive the gradient of $E(\theta)$ with respect to $\mu_j$. That is, compute $\frac{\partial E}{\partial \mu_j}$.

   (b) Derive the gradient of $E(\theta)$ with respect to $\pi_j$. That is, compute $\frac{\partial E}{\partial \pi_j}$. For now, ignore the constraints on the values of $\pi_j$.

   (c) A general way to handle constraints on parameters is to reparameterize the function in terms of unconstrained variables. In the case of the mixing proportions we have the constraint $\sum_{j=1}^{K} \pi_j = 1$. One way to reparameterize the mixing proportions is with the *softmax function*

$$\pi_j = s_j(w_1, \ldots, w_K) = \frac{\exp(w_j)}{\sum_{k=1}^{K} \exp(w_k)}$$

   where $\{w_j\}_{j=1}^{K}$ are the new parameters. Show that

$$\frac{\partial \pi_j}{\partial w_k} = \begin{cases} \pi_j(1 - \pi_j) & j = k \\ -\pi_j \pi_k & j \neq k \end{cases}$$

   (d) Use the results from (b) and (c) to derive the gradient of $E(\theta)$ with respect to $w_k$. Hint: Use the chain rule to combine the results to compute $\frac{\partial E}{\partial w_j}$.

2. We will now consider the first and second moments for a Gaussian Mixture Model.

   (a) Derive $E[y]$.

   (b) Derive $E[yy^T]$ and $\text{cov}[y] = E[(y - E[y])(y - E[y])^T]$.

# Dimensionality Reduction with PCA

In this assignment you will implement and apply dimensionality reduction for classification problems. The input you'll consider will be images of faces and you'll explore the problem of both face recognition (i.e., does an image contain a face) and person recognition (i.e., whose face is in this image).

**What to hand in:** As with the other assignments, you will submit a report with your results and analysis. In addition, you should also submit a ZIP file containing all of your source code implementing the methods.

## Step 1

Download the data from the Moodle. Included is a file called FACE_IMAGES.MAT which can be loaded in Matlab directly with the LOAD() command or in Python using the SCIPY.IO.LOADMAT function in the SciPy package. In that file will be four arrays in total. The first two, TRAIN_IMGS and TEST_IMGS, contain the images for the training and testing sets. The arrays will be $N \times 200 \times 180$ where $N$ is the number of images in the set. The other two arrays, TRAIN_IDS and TEST_IDS, are $N \times 1$ and contain a number for every image, indicating the identity of the person in that image.

To begin, display a few face images to make sure you know how to access the arrays. Next, compute and display the average of the training face images. This should look like a very blury face. See the IMSHOW() functions in Matlab and SciPy/Matplotlib for how to display the face images.

**What to include in your report:** Include the image of the mean face.

## Step 2

Next, do PCA on all the training images. Be sure to use SVD on the data matrix rather than explicitly forming the covariance matrix. To inspect the results of PCA you should produce the following plots and include them in your report:

- Scree Plot: This is a plot of the eigenvalues (note: not singular values) versus their index. Your eigenvalues should be sorted in decreasing order. See Figure 1.

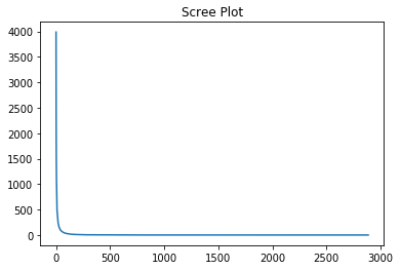



Figure 1: Scree Plot

- Fraction of Variance Plot: This is a plot of the fraction of variance explained versus the dimension of the subspace used. The class notes have the formal definition of fraction of variance explained. Note the CUMSUM() function in both Matlab and Python will be useful for computing this. See Figure 2.

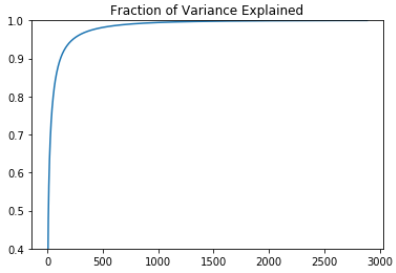



Figure 2: Fraction of Variance Explained

- Principle Components: Display (as images) the first 10 principle components corresponding to the largest eigenvalues. See Figure 3.

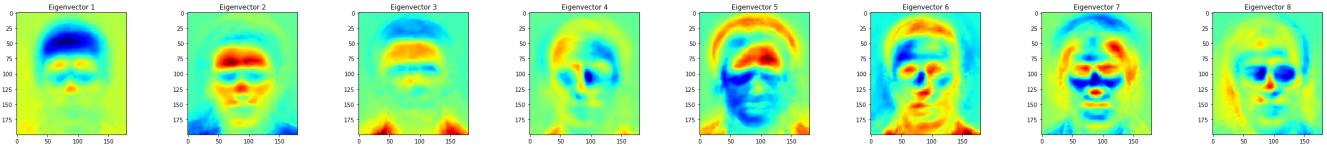


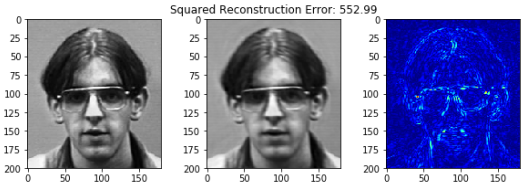

Figure 3: Eigenvectors. Red and blue correspond to regions of the image increasing and decreasing in intensity respectively. Green corresponds to regions that aren't changing in this direction.

Finally, select the dimensionality of the subspace to use. In particular, find the smallest dimensionality such that at least 90% of the variance is explained. **Note:** You should implement PCA yourself, directly calling SVD from your own code.

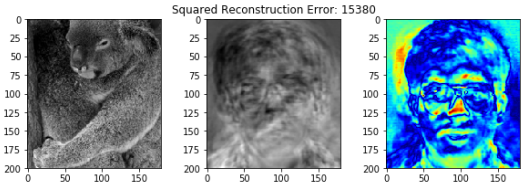
**What to include in your report:** Include the plots and images you produced. Report the subspace dimensionality that was selected to explain at least 90% of variance. And, for the first few principle components, try to explain what you're seeing in the principle components. What kind of variations in the images are being explained with those components?

## Step 3

Compute the subspace coefficients for all the training face images, using the dimensionality you determined in Step 2. For each image, use the subspace coefficients to reconstruct the expected image and compute its squared reconstruction error. That is, compute $\vec{y}_{recon} = \vec{b} + W\vec{x}$ where $\vec{b}$ is the mean image, $W$ are the basis vectors and $\vec{x} = W^T(\vec{y} - \vec{b})$ are the subspace coefficients for the original image $\vec{y}$. Then compute the squared reconstruction error $e = \|\vec{y}_{recon} - \vec{y}\|^2$. For a 5 randomly selected face images, display the original image, the reconstructed image and the absolute difference image and note the reconstruction error. It should look something like:




Load the images from the NONFACE_IMAGES.MAT and do the same thing, that is for each nonface image compute the reconstructed image and the squared reconstruction error. As above, for 5 randomly selected nonface images, display the original image, the reconstructed image and the absolute difference image and note the reconstruction error. It should look something like:




Finally, take the reconstruction errors for all the face images and non-face images and produce a histogram of the squared reconstruction errors. The histogram should allow you to see the differences of the errors between face and non-face images.

A simply way to determine if an image contains a face or not is to threshold the squared reconstruction error. If it is higher than a certain amount then it must be a nonface image. Use the histogram you generated to select a value for the threshold by hand. Use this value to classify the training face images and non-face images. Produce reconstruction images (as above) for face images which were misclassified as non-face images and non-face images which were misclassified as faces.

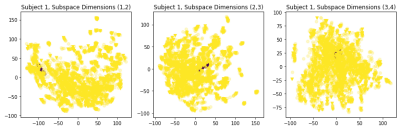
**What to include in your report:** Include 5 randomly selected face reconstructions and 5 randomly selected non-face reconstructions with errors annotated. Include the histogram you produced. Indicate on the histogram the selected threshold.

Try to explain why the images may have been misclassified. What particular features of the face and nonface images seem to be causing problems? Do you think these issues could be overcome with more data?

You should notice that the reconstructions of non-face images have a tendency to look like faces. Explain why this is the case. Why is there such a difference in the reconstruction quality between face and nonface images?

## Step 4

We are finally going to consider the problem of person recognition. That is, how can we determine if an image of a face contains a specific person. To begin we'll produce some plots of the subspace coefficients but colour coded based on subject identity. For the first 3 subjects, produce 2D scatter plots of pairs of subspace coefficients of all face images where the points are colour coded based on whether they are of the same person or a different person. For each subject produce plots of subspace dimensions (1,2), (2,3), and (3,4). For instance, for subject 1 the three plots should look roughly like this:




In these plots the dark points correspond to subject 1 and the yellow points correspond to other people. Note that you may have to play with the plotting settings a little bit to get things to look reasonable. They don't need to look exactly like this, but you should be able to get the idea.

These plots suggest that subject identities are well localized in the space, however simple linear decision boundaries may not work well due to crowding in the space. Instead, we're going to use a 1-nearest neighbours classifier. Implement the 1NN classifier using a 10 dimensional PCA subspace and test it using the test face images and compute the accuracy on the test set. **Note:** You should implement nearest neighbours yourself, that is, do not use a library or other builtin functions.

**What to include in your report:** Include subspace plots as described for two different subjects.

Report the 1NN classifier results. There should be a few errors on the test set (roughly 8). For each error on the test set produce images of the test image, the PCA subspace reconstruction of the test image, the 1NN training image and the PCA subspace reconstruction of the 1NN training image. Include these images in your report.

Discuss the error cases. Do they make sense when just looking at the original images? How about when you look at the reconstructions? Do you think increasing the subspace dimension would help with recognition performance? Justify your answer in terms of the results you have.

In this dataset the the backgrounds are consistently flat and the face occupies most of the image. How do you think the performance of this method would change if these things weren't true, e.g., the faces were only maybe 50% of the pixels in the image and the background was very cluttered and highly variable? Justify your answer in terms of how the PCA subspace would change. If you had more data would that change things?