Quantitative Methods in Neuroscience
(neu 466M)
Homework 7
Due: Thursday March 31 by 5 pm in NHB 3.350 (or by 2 in class if preferred)

This homework explores Principal Components Analysis (PCA) and some interesting uses. *General guidelines:* Read through each complete problem carefully before attempting any parts. Feel free to collaborate in groups of size 2-3, but always note the names of your collaborators on your submitted homework. For graphs: clearly label your axes and use good color and symbol choices. Print out your matlab code (in the form of a script file). For derivations you're asked to do 'by hand' (in other words, analytically, using paper and pencil) feel free to turn in handwritten or typed-out work.

1) **Spike sorting with PCA.** In many *in vivo* studies, action potentials of single neurons are recorded by extracellular electrodes. Action potentials are large enough events that the voltage fluctuations are visible to an electrode placed many micrometers away from the cell body. However, it is often the case that an electrode views action potentials from multiple cells at the same time. The waveforms from individual cells tend to have a stereotypical shape, and we can use this fact to cluster the waveforms from individual cells in order to identify the source of each action potential.

   The file `SpikeSortingData.mat` contains a sample dataset from a typical extracellular recording. The variable `voltageRecording` is the entire voltage sequence recorded during the experiment. The matrix `waveforms` contains all the action potential events extracted from the recording: each row of the matrix is the recorded waveform of an action potential. We can think about each time-point in the waveform as a separate variable. The shape of a waveform is determined by the covariance of different times within the waveform.

   a. Begin by plotting a few of the waveforms on the same plot. How many distinct neurons (based on the different waveform shapes) do you think can be seen in this recording?

   b. Calculate the covariance matrix of the waveforms using the `cov` command. The 70×70 covariance matrix represents how the different time-points in the waveforms vary relative to each other. Find the largest 20 eigenvalues and corresponding eigenvectors of this covariance matrix (use `eigs`). Plot the eigenvalues. How rapidly do the values fall off? It's often useful to plot the y-axis on a log scale. Also plot the cumulative sum of the ordered eigenvalues, normalized by the last point in the cumulative sum (i.e. by the sum over all 20 eigenvalues).

c. Plot the first four eigenvectors (these should each be of the same length as the individuals waveforms). The eigenvectors are "features" of the waveforms picked out as having the highest variance (largest eigenvalues) across the dataset. Interpret the three eigenvectors – what features of the actual waveforms do they approximate? When spike sorting by hand, it is common to pick out waveform height, number of lobes, and the depth of the lowest trough as features for differentiating between spikes from different cells. How do the PCA features compare?

d. For each data waveform, compute its coefficients (projection) onto the first and the second eigenvectors or features. This reduces the data waveform into a 2D coordinate, given by the two coefficients. Plot the coordinates for the different waveforms as a scatterplot – you should see clear clusters. What does this plot reveal about the number of neurons recorded in the experiment/the identity of the different waveforms? Explain. By contrast, try picking features by hand: e.g. pick timeindex 25 and time 30 (corresponding to the first peak and following trough) in the waveforms as the two coordinates and make a new scatterplot. Comment. Pick some different notable times or other features by hand and see if you can do better.

e. Finally, use the function `kmeans` to cluster the lower-dimension waveforms (try `c = kmeans(projectedWaveforms,k);` where `k` is the number of neurons you think are there). Now plot separately the waveforms as you did in part [a.], but only the waveforms for which `c=1` (Hint: try command `plot(1:70,waveforms(c==1,:))`). Make separate plots for `c=1` to `k`.

2) **Compression or dimensionality reduction with PCA.** Download the file `HandwrittenDigits.mat` and the function `plotImage.m` from the course website. The `.mat` file contains a matrix of images of handwritten numbers. Each row of the matrix contains a 28x28 pixel image, with elements resorted into a vector. The helper function `plotImage` will take a single row of this matrix as its argument and plot an image (for instance, after loading `HandwrittenDigits.mat`, type `plotImage(images(1,:))`). Have a look at a few different digit images.

a. Get the largest 200 eigenvalues and eigenvectors of the covariance matrix of the `images` data. Plot the eigenvalues and the cumulative sum of the eigenvalues normalized by the total sum of the eigenvalues. Use `plotImage` to plot the top 10 eigenvectors (try the `subplot` option so that you can put all these plots on the same page) and the bottom 10 eigenvectors. What is different about the eigenvectors associated with the largest versus the small eigenvalues?

b. Low-dimensional approximation of the images: Because the set of all eigenvectors

is a basis for the images, any image can be written as its projection (coefficient) onto the kth eigenvector times that eigenvector, summed over all k. Verify this for the first image $\mathbf{x}$ (the first row of the images matrix). Next, a low-dimensional representation of $\mathbf{x}$ can be generated by computing the sum over only the top few eigenvectors. Reconstruct a low-dimensional version of $\mathbf{x}$ using only the first two eigenvectors. Plot both the original and reconstructed image using `plotImage`. Now reconstruct $\mathbf{x}$ using the first 20 eigenvectors. The resulting vector should be the sum $\mathbf{x}^{low\ dim} = \sum_{\alpha=1}^{20} (\mathbf{x}^{\mathbf{T}} \mathbf{v}_\alpha) \mathbf{v}_\alpha$ where the $\mathbf{v}_\alpha$ are the eigenvectors and $\mathbf{x}$ is the original image.

c. Let's be systematic about the quality of the low-dimensional approximation: Calculate the reconstruction of the first image from the first k eigenvectors, as k varies from 1 to 200. Calculate the mean squared error (MSE) of each reconstruction (by taking the mean of the squares of the differences in pixel values between the original and reconstructed image vectors). Plot the MSE as a function of k. Do the same for a few different images from the images data set as well. Keeping in mind that the original images are each $28 \times 28 = 784$ dimensional vectors and that there are 10 different handwritten digit categories, what do these plots reveal about the data?