PSYCH-GA.2211/NEURL-GA.2201 – Fall 2020
Mathematical Tools for Neural and Cognitive Science

Homework 2

Due: 11 Oct 2020
(late homeworks penalized 10% per day)

See the course web site for submission details. Reminder: rather than using the functions pinv() and norm(), use the linear algebra tools we learned in class. Please: don't wait until the day before the due date... start *now*!

1. **Trichromacy.** Load the file `colMatch.mat` in your MATLAB environment. This file contains matrices and vectors related to the color matching experiment presented in class. In particular, the variable `P` is an $N \times 3$ matrix containing wavelength spectra for three "primary" lights, that could be used in a color-matching experiment. For these problems $N = 31$, corresponding to samples of the visible wavelength spectrum from 400nm to 700nm in increments of 10nm.

   The function `humanColorMatcher.p` simulates a normal human observer in a color matching experiment. The input variable `light` should contain the wavelength spectrum of a test light (a 31-dimensional column vector). The input variable `primaries` should contain the wavelength spectra of a set of primary lights (typically, a 31×3 matrix, as for matrix `P` described above). The function returns a 3-vector containing the observer's "knob settings" - the intensities of each of the primaries that, when mixed together, appear identical to the test light. The function can also be called with more than one test light (by passing a matrix whose columns contain 31-dimensional test lights), in which case it returns a matrix whose columns are the knob settings corresponding to each test light.

   (a) Create a test light with an arbitrary wavelength spectrum, by generating a random column vector with 31 positive components (use `rand`). Use `humanColorMatcher` to "run an experiment", asking the "human" to set the intensities of the three primaries in `P` to match the appearance of the test light. Compute the 31-dimensional wavelength spectrum of this combination of primaries, plot it together with the original light spectrum, and explain why the two spectra are so different, even though they appear the same to the human.

   (b) Now characterize the human observer as a linear system that maps 31-dimensional lights to 3-dimensional knob settings. Specifically, run a set of experiments to estimate the contents of a 3×31 color-matching matrix `M` that can predict the human responses. Verify on a few random test lights that this matrix exactly predicts the responses of the function `humanColorMatcher`.

   (c) The variable `Cones` contains (in the rows) approximate spectral sensitivities of the three color photoreceptors (cones) in the human eye: `Cones(1,:)` is for the L (long-wavelength, or *red*) cones, `Cones(2,:)` the M (green) cones, and `Cones(3,:)` the S (blue) cones. Applying the matrix `Cones` to any light $\vec{l}$ yields a 3-vector containing the average number of photons absorbed by that cone (try `plot(Cones')` to visualize them!). Verify that the cones provide a physiological explanation for the matching experiment, in

that the cone absorptions are equal for any pair of lights that are perceptually matched. First, do this informally, by checking that randomly generated lights and their corresponding 3-primary matching lights produce equal cone absorptions. Then, provide a few lines of matlab code that provide a more mathematical demonstration, along with an extended comment explaining your reasoning using concepts from linear algebra. [Hints for two possible approaches: (1) write math/code that computes cone responses for any test light and then computes the weighted combination of primaries that would produce the same cone responses - show that this is numerically the same as the color-matching matrix; (2) convince yourself, and explain why, it is sufficient to show that `M` and `Cones` have the same nullspace. Then use SVD to demonstrate that this is true!]

(d) The function `altHumanColorMatcher(light,primaries)` simulates a color-deficient human observer in a standard color matching experiment. (i) for a random test light, compare the knob settings for this observer with those of the normal human. Do this for several runs of `altHumanColorMatcher(light,primaries)`. How do they differ? (ii) Compute cone absorptions for the test light, and for the mixture of three matching primaries (by applying the `Cones` matrix). Do this for both the normal human observer, and for multiple runs of the abnormal observer. Try this for several different test lights. How do the cone responses of the normal and abnormal observers differ? Can you offer a diagnosis of the underlying cause of color deficiency in the abnormal observer?

2. **2D polynomial regression.** Load the file `regress2.mat` into your MATLAB environment. The matrix D contains 3 columns of data, which we'll refer to as X, Y, and Z respectively.

   (a) plot Z as a function of X and Y using `surf` [note: you'll need to reshape the three column vectors into square matrices]. Execute the command `rotate3d on`, and use the mouse to rotate the 3D space and view the data at different angles.

   (b) Fit the Z values with polynomials in X and Y, up to order 3: $p_0(X,Y) = \beta_0$, $p_1(X,Y) = \beta_0 + \beta_1 X + \beta_2 Y$, $p_2(X,Y) = \beta_0 + \beta_1 X + \beta_2 Y + \beta_3 X^2 + \beta_4 XY + \beta_5 Y^2$, etc. Compute this using `svd` and basic linear algebra manipulations that you've learned in class!

   (c) For each of the polynomials, (a) plot the fitted surface (use `surf`) and data points (use `plot3`) in the same figure, and rotate it around to convince yourself that the fit is reasonable. (b) compute the error for each element of Z, plot a histogram of these values, and compute the mean of the squared errors. How does the error behave as you increase the order of the polynomial? Which polynomial do you think gives the "best" fit? Explain.

3. **Trimmed regression.** One of the limitations of least-squares regression is sensitivity to outliers. A common solution is to iteratively discard the bad points. Load the file `regress3.mat`. First solve the standard regression problem, using a constant and a linear term. Then write a loop that locates the data point with the largest magnitude error (use MATLAB's `max`), eliminates that row from the data vector and regressor matrix, and then re-solves the regression problem. Note that you can also do this by including a diagonal weight matrix, and zeroing the entry corresponding to the outlier. On each iteration, plot the points and the best-fitting regression line, plot a histogram (use `hist`) of the squared errors over all points, and record the average over these errors (i.e., on the nth iteration, save the average error in the nth element of the vector). You may want put the two plots in separate figure windows, and you may want to pause after each iteration (use the `pause` function). Run the loop until half of the data points have been discarded.

After running, plot the vector of average errors, as a function of iteration, in a third figure. Based on this, and the histograms you observed, which iteration gave the "best" fit? To visualize this solution, plot the corresponding regression line, and *all* of the data points, labeling the discarded data points with a different plot symbol. Did you make a good choice?

4. **Dimensionality reduction with PCA.** Professors Hugh Bell and Wi Zell were recording extracellular action potentials (i.e. spikes) from cat primary visual cortex late one evening when their computer malfunctioned. It had already isolated a set of 400 time windows in which voltages had crossed a threshold, indicating the presence of spike. But these traces likely arose from multiple cells, with each cell producing a characteristic waveform, and the computer failed before sorting the voltage traces to determine how many cells were present, and which spikes arose from each cell. The professors come to you (the only math-tools-enabled graduate student still in the building at that hour), asking for help. They provide you with a file `windowedSpikes.mat` containing a 400×150 matrix, `data`, whose rows contain the electrode measurements (voltages recorded for each 150 msec window, at 1msec intervals). Your task is to determine how many neurons produced these 400 spikes.

   (a) Plot the 400 waveforms superimposed and describe what you see. Be sure to label your axes! Using these spike waveform plots, can you devise a way to deduce how many neurons produced these spikes? Feel free to include an additional plot containing just a subset of the waveforms in order to aid in your explanation.

   (b) Perform principal components analysis (PCA) on your data, and plot the eigenvalues in descending order (alternatively, compute the SVD of `data`). It might help to display the eigenvalues on a log-scale. Interpret what you see.

   (c) Measure the length of the projection of each of the 400 spike waveforms onto the top two principal components of the dataset, and plot the resulting values as points in 2 dimensions. Describe what you see. Can you deduce how many distinct neurons produced the 400 voltage traces?

   (d) Now project each waveform onto the top three principal axes, and plot in 3 dimensions (you may want to spin it around, using `rotate3d` in matlab). Are there any significant changes you see? Using the 3D plot, can you inform Drs. Bell and Zell how many neurons they likely recorded from?

   (e) Extra credit: how would you determine which neuron fired each of the 400 observed spikes? Describe your strategy, including appropriate linear algebraic expressions for the required calculations.