PSYCH-GA.2211/NEURL-GA.2201 – Fall 2024
Mathematical Tools for Neural and Cognitive Science

Homework 2

Due: 11 Oct 2024
(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 (or use `scipy.io.loadmat` for Python). This file contains matrices and vectors related to the color matching experiment presented in class. The variable `P` is an $N \times 3$ matrix containing wavelength spectra for three "primary" lights, to be used in a color-matching experiment. For these problems $N = 31$, corresponding to samples of the visible wavelength spectrum from 400 nm to 700 nm in increments of 10 nm.

   The function `humanColorMatcher.p` simulates a normal human observer in a color matching experiment. For Python, download the file and use `from trichromacy import human_color_matcher`. 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` in MATLAB or `np.random.rand` in Python). 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 `Phosphors` contains the emission spectra of three standard color display phosphors (from an old-fashioned cathode ray tube!). Suppose you wanted to make the background color of this screen match the appearance of an arbitrary test light. Write a matlab expression to compute the three phosphor intensities that would achieve this. Verify that this particular mixture of phosphor spectra satisfies the "matching" criterion (i.e., that a human would see this spectral mixture as being identical to the test light).

(d) The function altHumanColorMatcher(light,primaries) simulates a color-deficient human observer in a standard color matching experiment (for Python: from trichromacy import alt_human_color_matcher). (i) For a random test light, compare the knob settings for this observer with those of the normal human. Do this for several runs. How do they differ? (ii) 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 (for Python users, the indexing starts from 0). Applying the matrix Cones to any light $\vec{l}$ yields a 3-vector containing the average number of photons absorbed by each cone (try plot(Cones') to visualize them!). 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. **Polynomial regression.** Load the file regress1.mat into your MATLAB or Jupyter notebook environment. Plot variable Y as a function of X. Find a least-squares fit of the data with polynomials of order 0 (a constant), 1 (a line, parameterized by intercept and and slope), 2, 3, 4, and 5. [Compute this using svd and basic linear algebra manipulations that you've learned in class!] On a separate graph, plot the squared error as a function of the order of the polynomial. Which fit do you think is "best"? Explain.

3. **Constrained Least Squares Optimization.** Load the file constrainedLS.mat into a MATLAB or Jupyter notebook. This contains an $N \times 2$ data matrix, data, whose columns correspond to horizontal and vertical coordinates of a set of 2D data points, $\vec{d_n}$ (note that each $\vec{d_n}$ is a column vector but is a row of the matrix data). It also contains a 2-vector w. Consider a constrained optimization problem:

$$\min_{\vec{\beta}} \sum_n \left(\vec{\beta}^T \vec{d_n}\right)^2, \quad \text{s.t.} \quad \vec{\beta}^T \vec{w} = 1.$$

There is a family of possible vectors $\vec{\beta}$ that satisfy the *constraint* $\vec{\beta}^T \vec{w} = 1$. Geometrically, any $\vec{\beta}$ whose arrow-tip lies on a specific line perpendicular to $\vec{w}$ will satisfy the constraint. The perpendicular distance of this constraint line from the origin will be $1/||\vec{w}||$ from the origin (think about the dot product, draw the vector $\vec{w}$ and the constraint line to prove this to yourself). Thus, this is a new contrained optimization that is a bit like total least squares, except that β is forced to satisfy a linear constraint, rather than forced to be a unit vector.

(a) Rewrite the optimization problem in matrix form. Then rewrite the problem in terms of a new optimization variable, $\tilde{\beta}$ (i.e. 'beta tilde', a linear transformation of $\vec{\beta}$), such that the quantity to be minimized is now $||\tilde{\beta}||^2$. Note: you must also rewrite the constraint in terms of $\tilde{\beta}$.

(b) The transformed problem is one that you should be able to solve. In particular, you must find the shortest vector $\tilde{\beta}$ that lies on the constraint line. Compute the solution for $\tilde{\beta}$, and plot the solution vector, the constraint line and the transformed data points.

(c) Transform the solution back into the original space (i.e., solve for $\vec{\beta}$). Plot $\vec{\beta}$, the original constraint line, and the original data points. Is the optimal vector $\vec{\beta}$ perpendicular to the constraint line? On the same graph, plot the total least squares solution (i.e., the vector that minimizes the same objective function, but that is constrained to be a unit vector). Are the two solutions the same?

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 1 msec 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?