PSYCH-GA.2211/NEURL-GA.2201 – Fall 2023
Mathematical Tools for Neural and Cognitive Science

Homework 2

Due: 17 Oct 2023
(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. 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 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 `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

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. 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. The corresponding elements of these vectors, (X_k, Y_k, Z_k), represent 3D data points. X and Y are uniformly distributed on a square grid.

   (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, but the X and Y values lie on a square grid, so that isn't difficult]. 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. **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. **Principal components.** Load the file `PCA.mat` into your MATLAB environment. You'll find a matrix M containing responses of a population of 14 neurons, under 150 different experimental conditions (each column contains the estimated firing rate of one neuron under each of the conditions). The conditions correspond to an animal reaching for a target in a different directions, and different distances away from a central resting position. We cannot directly visualize data of this many dimensions, but we can use linear algebra to project them into a lower dimensional space.

(a) Compute the principal components of the 14-dimensional population responses (these will each be vectors with 14 components, i.e., one weight per neuron, thus treating conditions as 150 samples of 14-dimensional neural population responses). First, center the data by subtracting the mean response $mean(M)$ from every row of the matrix (hint: you might find the function `repmat` helpful). Call this re-centered data matrix $\tilde{M}$. Then compute the eigenvectors and eigenvalues of $\tilde{M}^T\tilde{M}$ (alternatively, you can compute the singular values of $\tilde{M}$). Plot the eigenvalues (or singular values). What do you think is the "true" dimensionality of the responses?

(b) Project the data in $\tilde{M}$ onto the first principal component (i.e., the eigenvector corresponding to the maximal eigenvalue). Plot a histogram (using `hist`) of these values. Verify that the sum of squares of these values is equal to the first eigenvalue λ_1. What proportion of the total variability of the data (the sum of squares of all entries of $\tilde{M}$) does this component account for?

(c) Show a scatter plot of the data projected onto the first two principal components (that is, plot the inner product of the data with the first component versus the inner product with the second component). You can use `plot` (with circular plot symbols and no connecting lines), or use `scatter`. Use `axis('equal')` to set the two axes to use equal scales. Show that the sum of the squared lengths of these projected vectors is equal to $\lambda_1 + \lambda_2$. What proportion of the total variability of the data do these two components account for?

(d) It appears that much of the response in this 14-neuron population can be explained in terms of these two components. Let's visualize this, by projecting the population response for each condition back onto these two axes, and plotting these as points in a 2D scatter plot.