PSYCH-GA.2211/NEURL-GA.2201 – Fall 2014
Mathematical Tools for Cognitive and Neural Science

Homework 2

Due: 10 Oct 2014
(late homeworks penalized 10% per day)

Save the solutions to each numbered problem as sections of a file called `hw2.m` in a folder called `hw2_Lastname`, along with additional files containing any functions you create. Send a zipped copy of this folder as an attachment in an email message with these attributes:

`To: catherio@nyu.edu, asr443@nyu.edu`
`Subject: Math Tools HW2`

Don't wait until the day before the due date... start *now*!

1. **Gram-Schmidt.** A classic method for constructing an orthonormal basis is known as *Gram-Schmidt orthogonalization*. First, one generates an arbitrary unit vector (e.g., by normalizing a vector created with `randn`). Each subsequent basis vector is created by generating another arbitrary vector, subtracting off the projections of that vector along each of the previously created basis vectors, and normalizing the remaining vector. You should draw (by hand) the picture in 2D, assuming you have one unit vector, and you're creating the second, to make sure you understand the geometry of this construction!

   Write a MATLAB function `gramSchmidt` that takes a single argument, N, specifying the dimensionality of the basis. It should then generate an $N \times N$ matrix whose columns contain a set of orthogonal normalized unit vectors. Try your function for $N = 3$, and plot the basis vectors (you can use MATLAB's `rotate3d` to interactively examine these). Check your function numerically by calling it for an N larger than 3 and verifying that the resulting matrix is orthonormal. Extra credit: make your function *recursive* – instead of using a `for` loop, have the function call itself. To do this, you'll probably need to write two functions.

2. **Trichromacy.** Load the file `colmatch.mat` in your MATLAB environment. This file contains a number of 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 basis lights (sometimes called "primaries"), and the variable `M` is a $3 \times N$ color-matching matrix corresponding to these primaries. For these problems $N = 31$, corresponding to samples of the wavelength spectrum from 400nm to 700nm in increments of 10nm.

   For any light $\vec{l}$, the product $M\vec{l}$ gives a 3-vector containing the intensities of the three primaries that, when combined, would appear the same as $\vec{l}$ to a human observer. That is, $P * M * \vec{l}$ appears the same as $\vec{l}$. More generally, two lights $\vec{l}_1$ and $\vec{l}_2$ will match in color appearance if and only if $M\vec{l}_1 = M\vec{l}_2$.

   (a) Create a light with a random wavelength spectrum, by generating a random column vector with 31 positive components (use `rand`). What combination of the three primaries in `P` will match the appearance of this randomly chosen test light? Compute the (N-dimensional) wavelength spectrum of this combination of primaries, and verify that it satisfies the general "matching" criterion described above. Plot it together

with the original light spectrum, and explain why the two spectra are so different, even though they would appear the same to a human.

(b) The variable `Phosphors` contains the emission spectra of three standard color display phosphors (it's an old-fashioned cathode ray tube!). Suppose you wanted to make the background color of this screen match the appearance of the light you generated in the previous problem. Write a matlab expression to compute the three intensities that should be for the phosphors. Verify that this mixture of phosphor spectra satisfies the "matching" criterion described above.

(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. 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, verify 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. [Hint: First convince yourself that it is equivalent to show that `M` and `Cones` have the same nullspace. Then use SVD to demonstrate that this is true]

3. **Polynomial regression.** Load the file `regress1.mat` into your MATLAB 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 [Note: compute this using `svd` and basic linear algebra]. 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.

   Trimmed regression. One of the limitations of least-squares regression is sensitivity to outliers. A common solution is to iteratively discard the bad points. First solve the standard regression problem. 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 basis function (regressor) matrix, and then re-solves the regression problem. On each iteration, record (in the entries of a vector) the mean squared error of the fit, which should steadily decrease. Now plot this error vector. On which iteration do you think the fit is "best"? For this iteration, plot the regression line, and the data points, labeling the discarded data points with a different plot symbol. Did you make a good choice?