CSE/NB 528 Homework 4: Learning in Neurons and Networks

Please turn in your solutions to these problems by midnight 11:59 PM on Monday, Feb 27, 2017.

Submission Procedure:
Create a Zip file called "528-hw4-*lastname-firstname*" containing the following:
(1) Document with write-up specifying the problems you are attempting, with your answers to any questions asked in the problem, as well as any figures, plots, or graphs supporting your answers,
(2) Your Matlab/Python program files,
(3) Any other supporting material needed to understand/run your solutions in Matlab/Python.
Upload your Zip file to this dropbox.
Upload your file by 11:59 PM Monday, Feb 27, 2017.

---

1. **Unsupervised Learning (100 points):** Write Matlab code to implement Oja's Hebb rule (Equation 8.16 in the Dayan & Abbott textbook) for a single linear neuron (as in Equation 8.2) receiving as input the 2D data provided in c10p1.mat but with the mean of the data subtracted from each data point. Note that this is a text file with 2 columns of data points.

   If using MATLAB: use "load –ASCII c10p1.mat" and type "c10p1" to see the 100 (x,y) data points. If using Python: use "data = np.loadtxt('c10p1.mat')". Compute and subtract the mean (x,y) value from each (x,y) point. Display the points again to verify that the data cloud is now centered around 0. Implement a discrete-time version (like Equation 8.7) of the Oja rule with $\alpha = 1$. Start with a random **w** vector and update it according to **w**(t+1) = **w**(t) + delta*d**w**/dt, where delta is a small positive constant (e.g., delta = 0.01) and d**w**/dt is given by the Oja rule (assume $\tau_W = 1$). In each update iteration, feed in a data point **u** = (x,y) from c10p1. If you've reached the last data point in c10p1, go back to the first one and repeat. Keep updating **w** until the change in **w**, given by *norm*(**w**(t+1) - **w**(t)), is negligible (i.e., below an arbitrary small positive threshold), indicating that **w** has converged.

   a. To illustrate the learning process, print out figures displaying the current weight vector **w** and the input data scatterplot on the same graph, for different time points during the learning process.
   b. Compute the principal eigenvector (i.e., the one with largest eigenvalue) of the zero-mean input correlation matrix (this will be of size 2 x 2). Use the matlab function "eig" to compute its eigenvectors and eigenvalues. Verify that the learned weight vector **w** is proportional to the principal eigenvector of the input correlation matrix (read Sections 8.2 and 8.3).