Laboratory Assignment 2 - 1 (LAB2-1)
Hebbian Learning

Solve the following assignment, whose completion is required to access the oral examination. Upload the assignments all-together in the Moodle platform of the course (once you have completed all the labs, not only this single one) as a compressed folder including one subfolder for each laboratory.

The subfolder for this lab should be called "LAB2_1" and should include the MATLAB scripts and the other files as requested in the assignment below. You can organize the code as you wish, implementing all the helper functions that you need, provided that these are included in the subfolder and are appropriately called in the scripts.
Properly organize the files requested for the different assignments into different sub-folders. E.g. "LAB2_1/Assignment1" for the first assignment, "LAB2_1/Assignment2" for the second assignment, etc.

Bonus track assignments are meant to be for those who finish early, but they are not formally required for completing the Lab Assignment.

Supporting material for this assignment is listed below:

Lectures' Slides
Hebbian Learning from lectures "Part 2 Lecture 1"

Matlab documentation
Matlab User's Guide https://www.mathworks.com/help/index.html
Matlab documentation using the `help` command

Assignment 1 – Basic Hebbian Learning

The assignment consists in the following points:

1) Load the data for this assignment from the file "lab2_1_data.csv"
   (rows are input dimensions, columns are different input patterns)
2) Implement a linear firing rate model (v = w * u)
3) Implement the basic Hebb rule
   - start with a randomly initialized weight vector (e.g. from a uniform distribution over [-1,1]);
   - at each epoch:
     - shuffle the input data sets (e.g., using the command randperm)
     - for each pattern in the dataset
       - present the pattern and compute the output
       - apply the Hebb learning rule (in the iterated, discrete map version, with learning rate; e.g. in the online version) $\mathbf{w}_{new} = \mathbf{w}_{old} + \eta\, \Delta\mathbf{w}$
   - stop learning if the weight vector stabilizes with respect to the values in the previous epoch (e.g. use a threshold value on the norm of $(\mathbf{w}_{new} - \mathbf{w}_{old})$), or if a maximum number of epochs have been performed;
4) Adapt the weights in **w** using the learning rule (implemented in the previous point)

P1) After training is completed plot a figure displaying (on the same graph) the training data points (points in the bidimensional space), the final weight vector **w** resulting from the learning process (e.g., using plotv) and the principal eigenvector of input correlation matrix **Q** (e.g., compute the correlation matrix, apply the eig() function and find the eigenvector associated with the maximum-eigenvalue – note: by default eig() does not always return the eigenvalues and eigenvectors in sorted order). Use a bolder line for displaying the principal eigenvector.

P2) Generate two figures plotting the evolution in time of the two components of the weight vector w (for this you will need to keep track of w (t) evolution during training). The plot will have time on the x axis, and the weight value on the y axis (provide a separate plot for each component of the weight vector). Also provide another plot of the evolution in time of the norm of the weight vector during. Note: *3 plots are required*.

The output of this assignment should then consist in the following data:

- The script .m file(s)
- The evolution of the weight vector (e.g., in a collective matrix), provided in .mat format
- The plot in point P1), provided in .fig or in .png format
- The plots in point P2), provided (individually, separately) in .fig and or .png formats

Assignment 2 – Oja Rule

Solve the same assignment as “Assignment 1”, but using the Oja learning rule (play a bit with the alpha values, then chose only one setting)
The same output as in “Assignment 1” is due.

Assignment 3 – Subtractive normalization Rule

Solve the same assignment as “Assignment 1”, but using the Subtractive normalization learning rule.

The same output as in “Assignment 1” is due.

Bonus Track Assignment 1 – BCM Rule

Solve the same assignment as “Assignment 1”, but using the BMC learning rule.

The same output as in “Assignment 1” is due.

Bonus Track Assignment 2 – Covariance Rule

Solve the same assignment as “Assignment 1”, but using Hebbian learning implemented using the covariance rule.

The same output as in “Assignment 1” is due.