

Computational Neuroscience

AY 2022-23

Laboratory Assignment 2 (LAB2) Hebbian Learning

General Information

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 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.

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

Detailed instructions for submissions are given in the lecture note ["Intro to the course"](#).

The recommended programming language is MATLAB, but you are allowed to use Python (e.g., with NumPy, or TensorFlow).

If you use MATLAB, you will provide:

- the source files as a collection of the scripts (.m files) used to solve the assignment, or as a live script (.mlx file) with all the necessary code.
- the output data as a collection of binary .mat files, or in a single .mat file.

If you use Python, you will provide:

- the source files in one of these forms:
 - a collection of the scripts (.py files) used to solve the assignment, with a main script that calls all the necessary other scripts;
 - a Jupyter notebook (.ipynb file) with all the necessary code.
- the output data as a collection of pickle or json files, or in a single pickle/json file
 - in this case it is required to include a script to load the saved data in the chosen format.

In both cases, the required figures can be provided in .pdf files or in a graphics format (e.g., png, jpg). When multiple images are required, you are allowed to generate a single file including multiple subplots (or pages, if appropriate).

Supporting Material

Lecture's 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

Python documentation

<https://www.learnpython.org/>

<https://numpy.org/>

<https://matplotlib.org/>

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., in MATLAB 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)
$$w_{new} = w_{old} + \eta \Delta w$$
 - 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 $w_{new} - w_{old}$), 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., in MATLAB 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 in MATLAB `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). Finally, provide another plot of the evolution in time of the norm of the weight vector during training. Note: in this point, *3 plots are required*.

The output of the assignment should then consist in the following

- The source code.

- The evolution of the weight vector, i.e., the value of the weights for each update (e.g., in a matrix or tensor).
- The 4 plots generated in P1 and P2.

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

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.

Bonus Track Assignment 3 – Dataset #2

Solve the same assignments (A1-3, BTA1-2) using the data from the file “dataset2_lab_2_1.csv”