Computational Neuroscience

AY 2022-23

Laboratory Assignment 3 (LAB3)
TDNN and RNN networks

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 "LAB3_2" 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, PyTorch, 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 nootebook (.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

Notes on Echo State Networks and Reservoir Computing (course lecture): file “Part3 Lecture 4.pdf” in the Moodle platform (section Lecture Notes).

Additional material for this assignment: file “Lab3_2-AdditionalMaterial.pdf” (containing info on initialization and training of ESNs)

Assignment 1: NARMA10 task

This task consists in predicting the output of a 10-th order non-linear autoregressive moving average (NARMA) system. Further information on this task can be found in the paper *Gallicchio, Claudio, and Alessio Micheli. "Architectural and markovian factors of echo state networks." Neural Networks 24.5 (2011): 440-456.*

The input of the system is a sequence of elements $x(t)$ randomly chosen according to a uniform distribution over [0, 0.5]. The output of the target system is computed as follows:

$$y(t) = 0.3\,y(t-1) + 0.05\,y(t-1)\sum_{i=1}^{10} y(t-i) + 1.5x(t-10)x(t-1) + 0.1$$

Given the input value $x(t)$, the task is to predict the corresponding value of $y(t)$.

- Import the dataset from the .csv file “NARMA10.csv” (available on the Moodle platform), where the first row represents the input and the second row represents the target output. Different columns represent different time-steps.
- Split the data into training (the first 5000 time steps), and test set (remaining time steps). Note that for model selection you will use the data in the training set, with a further split in training (first 4000 samples) and validation (last 1000 samples).
    - For the sake of problem understanding, you can try to first visualize the time-series data (using the matplotlib library in Python or the plot command in Matlab)

The assignment consists in:

1. Implement from the scratch the code required to initialize, run, train and evaluate an Echo State Network (in NumPy or Matlab). Your implementation should take into consideration relevant hyper-parametrization of the neural network (e.g., number of reservoir neurons, spectral radius, etc.)
2. Perform a model selection (e.g., by grid search or random search) on the values of the hyper-parameters identified in the previous point. Select on the validation set the best hyper-parametrization, as the one with the smallest Mean Squared Error (MSE).
3. Train the ESN model with the selected hyper-parametrization on the whole training set, and evaluate its MSE on the training set and on the test set.

   NOTE: For each hyper-parameterization that you consider, the performance (on training, validation and test sets) should be averaged over a number of reservoir guesses, i.e. different random instances of ESNs with the same values of the hyper-parameters (and potentially different random weights)

The output of the assignment should then consist in the following

- The source code.

    - include a specification of the set of hyper-parameters and the range of values considered for the model selection
- All the Echo State Network variables corresponding only to the selected hyper-parametrization. This includes all the weight matrices of the Echo State Network architecture, i.e., the input weight matrix, the reservoir recurrent weight matrix, the readout weight matrix and the used bias vectors.
- All the hyper-parameters values of the selected Echo State Network
- Training, validation and test Mean Squared Error.
- A plot with the target and output signals plotted together over time (i.e. on the X-axis there is time, on the Y-axis there is the signal value), one for the whole training data, one for the test data.

Bonus-track Assignment 1: Mackey-Glass 17 task

Solve the same assignment as in the previous exercise, this time referring to the Mackey-Glass (MG) 17 task (see details on this task in the previous lab assignment file).

Bonus-track Assignment 2: Custom Python implementation

Implement the Echo State Network as a custom TensorFlow/Keras or PyTorch model. Pay attention to the efficiency of the implementation (e.g., you don't need to re-calculate the states you have already computed). Consider the possibility of using scikit-learn models for the readout.
Solve the previous (and following) problems with the custom python code obtained in this way.

Bonus-Track Assignment 3: Sequential MNIST classification task

Solve the sequential MNIST classification problem with an ESN (see details on this task from previous lab assignment files).