Laboratory Assignment: Neural Networks for temporal data processing (Lab3-1)

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_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. "LAB3_1/Assignment1" for the first assignment, "LAB3_1/BonusTrack1", etc.

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

Useful documentation:
MATLAB on line documentation on Time Series and Dynamic Systems:
https://it.mathworks.com/help/deeplearning/time-series-and-dynamic-systems.html?s_tid=srchbrcm
MATLAB documentation using the help command (e.g. `help train`)
A brief recap of the process for creating, customizing and training neural networks is in the file "LAB3_1-AdditionalMaterial.pdf" (available on the Moodle platform).

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

$$d(t) = 0.3\, d(t-1) + 0.05d(t-1)\sum_{i=1,\dots,10} d(t-i) + 1.5\, u(t-10)u(t-1) + 0.1.$$

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

Import the dataset from the MATLAB file NARMA10timeseries.mat. You can find the input and target time-series respectively in the fields *input* and *target* of the imported structure. 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). Try to plot the tim- series data using the command `plot`.

- Create and train Time Delay Neural Networks (`timedelaynet`) and Recurrent Neural Networks (`layrecnet`), using different values of the hyper-parameters (size of the hidden layer, length of the input delay for timedelaynet, training function, learning parameters, number of training epochs, etc.). As regards the hyper-parameters, you can either (manually) choose a set of values to consider, either (systematically) use a grid (or random/Bayesian search).
- Select the best network hyper-parameters on the validation set, the hyper-parameterization with the smallest Mean Squared Error (MSE) (e.g. by using the command `immse`).
- Train the selected network on all the whole bunch of training data and evaluate the MSE of such network on the training set and on the test set.

Notes:

- You can use cell2mat for manipulating input and target data, and num2cell for opposite data structure transformation.

The output of the assignment should then consist in the following data, pertaining to the Time Delay Neural Network and to the Recurrent Neural Network, only to the selected hyper parametrization:

- The script .m file(s)
  - This file should also include a specification of the set of the hyper-parameters values considered for the model selection (if not already in the grid)
- The net structure
- The TR (training record) structure
- Training, validation and test Mean Squared Error (e.g. by using `immse` command)
- A plot (in .fig or .png format) 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; use the hold on command to have a comparative plot). This type of plot is required for both training and test (2 plots).
- A plot of the learning curve (in .fig or .png format). This consists in a plot containing – at least – the training error computed over the training epochs.

Bonus-Track Assignment 1 – Laser Task

The Laser task consists in a next-step prediction (autoregressive, a particular case of transduction) on a time series obtained by sampling the intensity of a far-infrared laser in a chaotic regime.

Import the dataset from Matlab (`load laser_dataset`), rescale values to [-1,1], properly separate input and target data, then split the available data in training (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). Try to plot the time series data using the command `plot`.

- Create and train Time Delay Neural Networks (`timedelaynet`) and Recurrent Neural Networks (`layrecnet`), using different values of the hyper-parameters (size of the hidden layer, length of the input delay for timedelaynet, training function, learning parameters, number of training epochs, etc.). As regards the hyper-parameters, you can either (manually) choose a set of values to consider, either (systematically) use a grid.
- Select the best network hyper-parameters on the validation set, the hyper-parameterization with the smallest Mean Squared Error (MSE) (e.g. by using the command `immse`).
- Train the selected network on all the training data and evaluate the MSE of such network on the training set and on the test set.

Notes:

- You can use cell2mat for manipulating input and target data, and num2cell for opposite data structure transformation. E.g.
  load laser_dataset; %load the laser dataset
  allData = cell2mat(laserTargets); %converts the cell structured data into a row matrix.

- For next -step prediction tasks input and target data can be separated by an appropriate shift of indexing, e.g. `inputData = allData(1:end-1); targetData = allData(2:end);`
- To give a practical insight into the next-step prediction type of tasks, suppose that the input sequence is as follows:

time step	*1*	*2*	*3*	*4*	*5*	…
input value	3	5	9	12	8	…

  then: the desired output (i.e. the target) at step 1 is 5, at step 2 is 9, at step 3 is 12, and so on…

The output of the assignment should then consist in the following data, pertaining to the Time Delay Neural Network and to the Recurrent Neural Network, only to the selected hyper parametrization:

- The script .m file(s)
    - This file should also include a specification of the set of the hyper-parameters values considered for the model selection (if not already in the grid)
- The net structure
- The TR (training record) structure
- Training, validation and test Mean Squared Error (e.g. by using `immse` command)
- A plot (in .fig or .png format) 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; use the hold on command to have a comparative plot). This type of plot is required for both training and test (2 plots).
- A plot of the learning curve (in .fig or .png format). This consists in a plot containing – at least – the training error computed over the training epochs.

Bonus Track Assignment 2– Mackey-Glass Task

The Mackey–Glass (MG) time series is a standard benchmark for chaotic time series prediction models. The task of interest for this assignment is a next-step prediction task (autoregressive, a particular case of transduction) on the MG time series.
Import the dataset from the attached file "MGtimeseries.mat" (available on the Moodle platform in the "LAB3_1" folder), properly separate input and target data, then split the available data in training (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). Try to plot the time series data using the command `plot`.

- Create and train Time Delay Neural Networks (`timedelaynet`) and Recurrent Neural Networks (`layrecnet`), using different values of the hyper-parameters (size of the hidden layer, length of the input delay for timedelaynet, training function, learning parameters, number of training epochs, etc.). As regards the hyper-parameters, you can either (manually) choose a set of values to consider, either (systematically) use a grid.
- Select the best network hyper-parameters on the validation set, the hyper-parametrization with the smallest Mean Squared Error (MSE), e.g. by using command `immse`.
- Train the selected network on all the training data and evaluate the MSE of such network on the training set and on the test set.

Notes:

- You can use cell2mat for manipulating input and target data, and num2cell for opposite data structure transformation.
- For next -step prediction tasks input and target data can be separated by an appropriate shift of indexing, e.g. `inputData = allData(1:end-1); targetData = allData(2:end);`
- To give a practical insight into the next-step prediction type of tasks, suppose that the input sequence is as follows:

time step	*1*	*2*	*3*	*4*	*5*	…
input value	3	5	9	12	8	…

  then: the desired output (i.e. the target) at step 1 is 5, at step 2 is 9, at step 3 is 12, and so on…

The output of the assignment should then consist in the following data, pertaining only to the selected hyper parametrization:

- The script .m file(s)
- The net structure
- The TR (training record) structure
- Training, validation and test Mean Squared Error (`immse` command)
- Target vs output plot, both for training and test data (in .fig or .png format)
- A plot of the learning curve (in .fig or .png format)

Bonus Track Assignment 3– NARX networks

Apply NARX networks with different values of the hyper-parameters (size of the hidden layer, length of the input delay, length of the output delay, training function, learning parameters, number of training epochs, etc.) to the previous tasks in Assignment1 and Bonus Track Assignment1.

Bonus Track Assignment 4 – Long-Short Term Memory Networks (LSTM).

This tutorial is meant to help you familiarize with a popular model for sequential data, that is the LSTM.
Prerequisites: Install the Deep Learning Toolbox from the Add-ons.

We will learn how to predict the human activity (running, dancing, walking etc.) using accelerometer's data. There are a total of 6 training time-series, each of which has 3 features per time step (x,y,z axes) .

- You may want to plot the data to understand what you are dealing with. Use the following code:

```
load HumanActivityTrain;
load HumanActivityTest;

human_id = 1
feature = 1
X_tr = XTrain{human_id}(feature,:);
classes = categories(YTrain{human_id});

figure
for j = 1:numel(classes)
    label = classes(j);
    idxs = find(YTrain{human_id} == label);
    hold on
    plot(idxs, X_tr(idxs))
end
hold off
xlabel("Time Step")
ylabel("Acceleration")
title("Training Sequence" + string(human_id) + " Feature " + string(feature))
legend(classes,'Location','northwest')
```

- Have a look at [1]. To construct a neural network, we have to specify its layers.

```
numFeatures = 3;
numClasses = 5;

numHiddenUnits = CHOOSE IT;

layers = [ ...
    sequenceInputLayer(numFeatures) % specify the input dimension at each time step
    lstmLayer(numHiddenUnits,'OutputMode','sequence') % seq-to-seq prediction
    fullyConnectedLayer(numClasses) % add a dense layer (5 output neurons)
    softmaxLayer % interpret the output as probabilities
    classificationLayer];  % used to compute the classification loss
```

NOTE: if you want **(justify it)**, you can use a bidirectional-LSTM. Just replace lstmLayer with bilstmLayer.

- Specify the training options and launch training. We will use the last 2 training inputs as validation set to detect overfitting behaviors.

```
options = trainingOptions('adam', ...
    'MaxEpochs', CHOOSE IT, ...
    'InitialLearnRate', CHOOSE IT, ...
    'L2Regularization', CHOOSE IT, ...
    'ValidationData',{XTrain(5:6), YTrain(5:6)}, ...
    'ValidationFrequency', 1, ...
    'Plots','training-progress');

net = trainNetwork(XTrain(1:4), YTrain(1:4), layers, options);

YPred = classify(net,XTest); % inference on test set

acc = sum(YPred == YTest)./numel(YTest) % plot accuracy on test set
```

- Choose the best hyper-parameters on the validation set using a grid/random search.
- Once you have found them, report the final test accuracy. You cannot repeat the process to see if your test accuracy improves. The test set must be seen only one time, that is after model selection.

The output of this assignment must contain:

- The script .m file(s)
- A readme file with the best hyperparameters configuration with training, validation and test accuracy.

Note: Do not cheat (do not repeat the process because you got a worse test score than others, and choose the hyper-parameters to try by yourself).

Useful documentation

[1] Matlab LSTM: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html
[2] Sequence-to-Sequence Classification Using Deep Learning (this tutorial):
https://www.mathworks.com/help/deeplearning/examples/sequence-to-sequence-classification-using-deep-learning.html

Other resources:

[3] Time Series Forecasting Using Deep Learning:
https://www.mathworks.com/help/deeplearning/examples/time-series-forecasting-using-deep-learning.html
[4] Sequence Classification Using Deep Learning:
https://www.mathworks.com/help/deeplearning/examples/classify-sequence-data-using-lstm-networks.html