Exercise in Advanced Machine Learning
Prof. Dr. Youness Dehbi
Computational Methods Lab
HafenCity University Hamburg

# Exercise 3: Long Short-Term Memory

Hand-out : Wednesday, 03.05.2023
Hand-in : Wednesday, 31.05.2023

## General Notice

The exercises can be solved in groups of two to three students.Answers to theoretical questions and the source code of the programming exercises and result files should be presented on demand.

## Recurrent Neural Networks

Just as people do not have to think again each time about the things they have already learned, it is also possible to teach neural networks to recall knowledge they were being taught. This is done in so-called Recurrent Neural Networks (RNNs) with loops inside, which allow information to be retained. Currently the most used architectures of RNNs are Long short-term memory (LSTM) networks. LSTMs are RNNs that overcome the problem of long-term dependencies and thus have achieved the most state-of-the-art results in this area. In this exercise we will look at how to use LSTMs to predict future values using time series data sets.
In the source code, gaps, which are marked with **TODO****, have to be filled and theoretical questions have to be answered. For experimentation, further changes to the code should be made at the end, if requested. The following questions guide you step by step through the code:

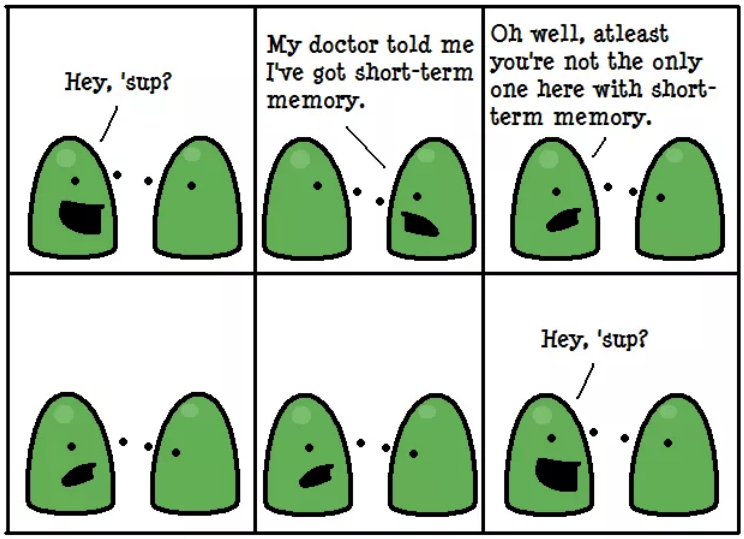


Figure 1: Short-term memory

### Read data

1. **(5)** You are provided with the tide gauge data for Pellworm in Schleswig-Holstein, Germany. The data is stored in the file `pellworm_data.csv`, where each row contains the information `[yyyy mm dd hh mm ss height]`. Read in the data set (like in previous exercises) and plot the height against time. Since the data set is very large, it is sufficient for this exercise to use only every 1000th height value.

2. **(4)** In addition, load the flight dataset, which is part of the seaborn library. Plot here the number of passengers against time.

3. **(4)** Explain briefly the characteristic differences between the time series (periodicity?, trends?).

We assume that the time series are equidistant, i.e. that all data points have the same distances. This means that we only need height resp. passenger data for the following steps. Decide which data set you want to use in this exercise.

## Data Preprocessing

Since we want to predict future data points in this exercise, we first cut our time series at the end and treat this cut-off part as test data.

4. **(2)** Set a suitable size for the `test_data` so that enough data remains for training and validation (`trva_data`).

The `trva_data` are now normalized and converted to torch tensors, as known from the previous exercises.
However, the next step is special by creating sequences from the `trva_data`. Therefore we use the helper function create_inout_sequences(input_data, train_window), which creates sequences $Input \rightarrow Output$ from the time series like this (in case `train_window = 3`):

$$[x_{t-2}, x_{t-1}, x_t] \rightarrow x_{t+1}, \tag{1}$$

The input sequences have the length `train_window` and the output has the length 1.

5. **(2)** Define a suitable `train_window` size.

In the following, the sequences are divided into training and validation sequences according to the previous exercises.

## Training

You are provided with the class LSTM. A look at the forward method shows that the input flows directly into LSTM modules. The number can be determined via `num_lstm_layers` and their size through `hidden_dim`. Significantly, in addition to the output, the lstm modules also have a hidden cell state. The last LSTM layer is followed by a linear output layer. Note that we use a linear output layer as in regression, because the output values are unbounded.

6. **(5)** Define the parameters of the network appropriately. Note that with an increasing number of neurons and LSTM modules the computational speed decreases.

7. **(4)** Set initial values for the hyper parameters, loss function and optimizer.
   Hint: MSELoss() and Adam optimizer are well suited.

8. **(5)** The training loops over the epochs and the individual sequences are already implemented. But the code is completely uncommented. Just add short one-line comments to each line in the loop.

## Testing

With the trained LSTM model we can now make predictions about future values. However, the model always expects as input a sequence of the length of `train_window`. To predict point $x_{t+1}$ we can simply take the last `train_window` values of the `trva_data`. To predict point $x_{t+2}$ we have to delete the first `trva_data` point and insert the previous test point $x_{t+1}$ at the end. This is repeated as often as desired in `fut_pred`.
Note that this procedure is also well suited for other tasks, for example to close data gaps.

9. **(4)** Change the value for `fut_pred` and describe how the values behave which are long in the future. Save the corresponding plot.

10. **(5)** Optimize the parameters of the network in such a way that the `test_data` are satisfactorily approximated. Crucial parameters are `train_window`, `hidden_dim`, and `num_lstm_layers`. Save this state for submission.