Time Series Analysis & Recurrent Neural Networks

lecturer: Daniel Durstewitz
tutors: Manuel Brenner, Daniel Kramer, Leonard Bereska
SS2020/2021

Exercise 10

To be uploaded before the exercise group on June 16th, 2021

Time series prediction with RNN and LSTM networks in PyTorch
In this exercise, we will compare a simple recurrent neural network (RNN) with an LSTM for time series prediction. The code in `main.py` (together with `train_rnn.py` and `train_lstm.py`) establishes a paradigm for training two neural networks (`MyRNN` and `MyLSTM`) for time series prediction. `MyRNN` and `MyLSTM` respectively are a single hidden layer RNN using ReLU activation and a single hidden layer LSTM network projecting into a linear output layer. Both networks have input and output dimension equal to one (the time series we want to predict in this exercise is uni-dimensional). We chose the size of the hidden layer so that both RNNs have an equal number of parameters.
In `data.npy` you will find a uni-dimensional time series generated by the Lorenz system equations (`https://en.wikipedia.org/wiki/Lorenz_system`). The first part of the code loads the time series, and formats and divides it into a training and test set. The training procedure is described in `train_rnn.py` and `train_lstm.py`. To train for one step ahead prediction, we divide the training set into intervals of length T and pack the data into mini-batches. As target output we use the same time series as provided as input shifted by one time step.

1. Train both networks separately by running `main.py` and choosing either `MyRNN` or `MyLSTM`, respectively. You can see how the network progressively improves its approximation of the target time series by looking at the figures automatically saved in your folder every 50 epochs. When training is finished, the error over training epoch will be plotted, together with a plot of the predicted and target time series.

2. To assess the prediction error of a network, we have to test it on data not used for the training procedure. Run the two trained models on the test set and plot, as in 1.), the predicted and target time series.

3. The two RNNs have been trained to predict one step. However, by iteratively applying the RNNs to their own outputs, we can also predict multiple steps into the future without having to re-train the networks. Compute and plot the Mean Squared Error (MSE) for n steps into the future, with n from 1 to 40, and compare the results for `MyRNN` and `MyLSTM`.

   To compute e.g. two steps into the future randomly select an interval $X_{1:T}$ of length T from your test set; apply the RNN a first time obtaining Y' (prediction of the true $Y := X_{2:T+1}$); concatenate the last element of the vector Y' to the original $X_{1:T}$ ($X'_{1:T} := [X_{1:T}; Y'(last)]$); apply the RNN to $X'_{1:T}$ obtaining Y'' ; the last element of Y'' corresponds to the two steps ahead prediction. Repeat the procedure for multiple intervals randomly selected from the test set and compute the MSE only on the predicted last elements.