Time Series Models: From Statistics to AI

lecturer: Daniel Durstewitz
tutors: Lukas Eisenmann, Christoph Hemmer, Alena Brändle, Florian Hess, Elias Weber, **Florian Götz**
SS2025

Exercise Sheet 8

Regulations

Please submit your solutions via Moodle in teams of **2 students**, before the exercise group on **Wednesday, June 18th, 2025**. Each submission must include **exactly** two files:

- A `.pdf` file containing both your Jupyter notebook and solutions to analytical exercises. The Jupyter notebook can be exported to pdf by selecting **File → Download as → pdf** in JupyterLab. If this method does not work, you may print the notebook as a pdf instead. Your analytical solutions can be either scanned handwritten solutions or created using LaTeX.
- A `.ipynb` file containing your code as Jupyter notebook.

Both files must follow the naming convention:
`Lastname1-Lastname2-sheet08.pdf`
`Lastname1-Lastname2-sheet08.ipynb`

1 Intuition on DSR with PLRNNs

Assume we have a piecewise-linear recurrent neural network (PLRNN) of the form

$$\mathbf{z}_{t+1} = p(\mathbf{z}_t) = \mathbf{A}\mathbf{z}_t + \mathbf{W}\phi(\mathbf{z}_t) + \mathbf{h} \tag{1}$$

and an observation layer

$$\mathbf{x}_t = \mathcal{I}\mathbf{z}_t \tag{2}$$

where $\phi(x) = \max\{0, x\}$ is the rectified linear unit (ReLU) function, applied to vectors element-wise, $\mathbf{A}$ is a diagonal matrix, $\mathbf{W}$ an off-diagonal matrix, $\mathbf{z}_t \in \mathbb{R}^M$, $\mathbf{x}_t \in \mathbb{R}^N$, $N \leq M$ and

$$\mathcal{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{N \times M} \tag{3}$$

That means that $\mathbf{x}_t$ is just a copy of the first N entries of $\mathbf{z}_t$. We say that the PLRNN $p(\mathbf{z})$ *reconstructs* an N-dimensional dynamical system $f(x)$, if for any initial state $\mathbf{z}_0 \in \mathbb{R}^M$, the time series $\mathbf{x}_0 = \mathcal{I}\mathbf{z}_0$, $\mathbf{x}_1 = \mathcal{I}\mathbf{z}_1 = \mathcal{I}p(\mathbf{z}_0)$, $\mathbf{x}_2 = \mathcal{I}\mathbf{z}_2 = \mathcal{I}p(\mathbf{z}_1) = \mathcal{I}p^2(\mathbf{z}_0)$, etc. behaves in accordance with f. That is:

- If f has an attracting fixed point $\mathbf{y}$ in the vicinity of $\mathbf{x}_0$, then $\mathbf{x}_t \to \mathbf{y}$ as $t \to \infty$.

- If $\mathbf{x}_0$ is on a cycle of f, there is a $t > 0$ such that $\mathbf{x}_t = \mathbf{x}_0$.
- If f is chaotic, so is the trajectory of $\mathbf{x}_t$, and so on.

A PLRNN can theoretically reconstruct any system, but its capacity to do so depends on its dimensionality M. However, this is a trade-off: computation time increases with M. Let $N = 1$. What is a lower bound to the PLRNN dimension M required to reconstruct a dynamical system f with the following properties? (No exact mathematical argument is required – just intuition; there is no definite answer to the questions.)

- f diverges.
- f has 1 fixed point.
- f has m fixed points.
- f has a cycle of order 2.
- f has a cycle of order $c > 2$.
- f is chaotic.

Hint: the answer has to do with the name "PLRNN". Where is it linear, exactly?

2 Training on chaotic data

The periodic sinusoidal data from the last exercise sheets were relatively easy to learn and reconstruct with a standard RNN – cycles are not a particularly big challenge for these. Now let us move on to a more interesting application.

The files `lorenz63_*.npy` contain data sampled from the Lorenz system,

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= \rho x - xz - y, \\ \dot{z} &= xy - \beta z, \end{aligned} \tag{4}$$

for $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$. For these parameters, the system exhibits chaotic behaviour.

a) Load the data using `numpy.load` and visualize it with a 3D plot.

We want to train on this dataset using various models: a standard RNN, a PLRNN and an ALRNN.

b) Use the standard RNN, which you have already used for the last exercises, and try to reproduce the attractor. How well does it work, judged by visual inspection and mean-squared error? *Hint/remark:* as the data are much more complicated than the previous simple sinusoidal dynamics, you will need a larger network (i.e., more hidden units) and train for more episodes than before. You may consider using L1-regularization with an appropriate regularization strength as well as mini-batching and the Adam optimizer for improved performance. Choose an appropriate network size and sequence length.

c) Implement the piecewise-linear RNN,

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \mathbf{W}\phi(\mathbf{z}_{t-1}) + \mathbf{h}, \tag{5}$$

where diagonal $\mathbf{A} \in \mathbb{R}^{M\times M}$ contains linear self-connections, $\mathbf{W} \in \mathbb{R}^{M\times M}$ are nonlinear connections between units, $\mathbf{h} \in \mathbb{R}^M$ is a bias term, and $\phi(z) = \max[0, z]$ is an element-wise ReLU nonlinearity. Use an identity observation model,

$$\hat{\mathbf{x}}_t = \mathcal{I}\mathbf{z}_t, \tag{6}$$

where $\mathcal{I} \in \mathbb{R}^{N\times M}$ with $\mathcal{I}_{rr} = 1$ for the N read-out neurons, $r \leq N$, and zeroes elsewhere (see also Exercise 1). During training, replace the read-out states with observations every τ time steps:

$$\mathbf{z}_{t+1} = \begin{cases} F_\theta(\tilde{\mathbf{z}}_t) & \text{if } t = \ell\tau + 1, \ell \in \mathbb{N}_0 \\ F_\theta(\mathbf{z}_t) & \text{else,} \end{cases} \tag{7}$$

where $\tilde{\mathbf{z}}_t = (\mathbf{x}_t, \mathbf{z}_{N+1:M,t})^\top$. This variant of sparse teacher forcing is called *identity teacher forcing*.

Note: There is a Jupyter notebook attached to this exercise sheet with much of the basic structure already provided, so you will mainly need to fill in some technical details and not implement the entire training from scratch. However, be sure to familiarize yourself with the code and understand how it works in order to get the maximum benefit from this exercise. Also note that the notebook does not serve as a full template for this exercise with all of its subtasks.

d) Now try to reduce the number of ReLUs while keeping the total latent dimension fixed, i.e., train using the almost-linear RNN architecture,

$$\mathbf{z}_t = \mathbf{A}\mathbf{z}_{t-1} + \mathbf{W}\Phi^*(\mathbf{z}_{t-1}) + \mathbf{h}, \tag{8}$$

where

$$\Phi^*(\mathbf{z}_t) = [z_{1,t}, \ldots, z_{M-P,t}, \max(0, z_{M-P+1,t}), \ldots, \max(0, z_{M,t})]^\top \tag{9}$$

and $0 \leq P \leq M$. How low can you go with P and still obtain good results?

e) Test the sensitivity of your trained model to small changes in the initial latent state $\mathbf{z}_0$. Does it reproduce the hallmark sensitivity of chaotic systems?

If you are having fun at this point, you may additionally continue to work on the following tasks (not mandatory!):

f) *Bonus:* Try adding noise to the data. How much noise can you add and still obtain a good reconstruction?

g) *Bonus:* Play around with the code and, for example, try generalized teacher forcing (GTF) instead of STF, or experiment with other kinds of RNN model and combinations with training techniques.