# Time Series Analysis & Recurrent Neural Networks

lecturer: Daniel Durstewitz
tutors: Manuel Brenner, Max Thurm, Janik Fechtelpeter
SS2022

## Exercise 11

To be uploaded before the exercise group on July 13, 2022

General notice: I noticed last time that there were a couple of very similar, yet individual solutions. It's a great idea to work together on a sheet, but then it would be helpful to hand it in together, too. This saves me some correction time and will result in the same number of points :)

**Task 1: Training on Chaotic Data** The periodic sinusoidal data from the last exercises was relatively easy to reconstruct - cycles are no problem for an RNN. Now to something more difficult. The dataset `lorenz63.tc` contains data sampled from the Lorenz system

$$\begin{aligned}\dot{x} &= s(y - x)\\ \dot{y} &= rx - xz - y\\ \dot{z} &= xy - bz\end{aligned}$$

for $s = 10, r = 28, b = \frac{8}{3}$. At this parameter configuration, the system is chaotic.

a) The provided script `rnn.py` provides a model definition, a train function, and plotting functions. Use the plot3d function to plot the data.

b) The model definition in `rnn.py` is the same as in the last exercises. Use the train function to train it on the Lorenz data. Note that I have changed a few details to make things easier. The train function employs L1 regularization, mini-batching and the Adam optimizer. Choose sensible parameters for regularization strength, batch size, and sequence length, and explain your reasoning*. Because the data is much more complicated than sine, you have to choose a larger number of hidden units and also train for a longer time (much more episodes than before). That's why it makes sense to test your code first with a smaller number of episodes, and then, if everything works, give it a couple of minutes until the loss converges. Note that the template plots the loss in the logarithmic domain, to observe it more easily.

c) Take a sub-sequence of 500 time steps from the data. Use the plot_generated function to predict a time series of the same length with the model and plot the two next to each other to judge whether the RNN did reconstruct the Lorenz system. Calculate the mean squared error between the two.

d) It sucked? It's supposed to. What is a possible reason for this?

e) Bonus: Try some tweaks to make prediction better. E.g. instead of the standard RNN, use an LSTM or GRU. Whoever minimizes the mean squared error between the first 500 time steps and a generated sequence of the same length, wins!

*Hint: sequence length has to do with the structure of the data, and also with calculation time

**Task 2: Reconstructing Dynamical Systems with PLRNNs** Assume we have a piece-wise linear recurrent neural network (PLRNN) of the form

$$z_{t+1} = p(z_t) = Az_t + W\phi(z_t) + h$$

and an observation layer

$$x_t = Bz_t$$

where $\phi(x) = \max\{0, x\}$ is the rectified linear unit (ReLU) function (the max is applied element-wise). $A$ is a diagonal, and $W$ an off-diagonal matrix, $z_t \in \mathbb{R}^n$, $x_t \in \mathbb{R}^k$, $k \leq n$ and

$$B = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{k \times n}$$

That means that $x_t$ is just a copy of the first $k$ entries of $z_t$. We say that the PLRNN $p(z)$ "reconstructs" a $k$-dimensional dynamical system $f(x)$, if for any initial state $z_0 \in \mathbb{R}^n$, the time series $x_0 = Bz_0, x_1 = Bz_1 = Bp(z_0), x_2 = Bz_2 = Bp(z_1)$, etc. behaves in accordance with $f$. That is, if $f$ has an attracting fixed point $y$ in the vicinity of $x_0$, $x_t \to_{t\to\infty} y$. If $x_0$ is on a cycle of $f$, there is a $t > 0$ such that $x_t = x_0$. If $f$ is chaotic, so is the trajectory of $x_t$, and so on.

A PLRNN can theoretically reconstruct any system, but its capacity to do so depends on its dimensionality $n$. However, this is a trade-off: computation time grows with $n$. Let $k = 1$. What is a lower bound to the PLRNN dimension $n$ required to reconstruct a dynamical system $f$ with the following properties? (no exact mathematical argument 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?