Dynamical Systems Theory in Machine Learning & Data Science

Lecturers: Daniel Durstewitz
Tutors: Christoph Hemmer, Alena Brändle, Lukas Eisenmann, **Florian Hess**
WS2024/25

Exercise 11

To be uploaded before the exercise group on January 22, 2025

1 Reservoir Computing

In this exercise we want to explore the Reservoir Computing (RC) paradigm for DSR. To this end, we will implement an Echo-State Network (ESN). ESNs can be seen as RNNs with fixed dynamical weights and a trainable linear observation model. The ESN we will look at takes the form

$$\begin{aligned} \boldsymbol{r}_t &= (1-\alpha)\boldsymbol{r}_{t-1} + \alpha \tanh(\boldsymbol{W}\boldsymbol{r}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{b}) \\ \hat{\boldsymbol{x}}_t &= \boldsymbol{W}_{out}\boldsymbol{r}_t \end{aligned} \tag{I}$$

where $\theta = \{\alpha, \boldsymbol{W}, \boldsymbol{W}_{in}, \boldsymbol{b}\}$ are drawn randomly and stay fixed during training. To train the ESN, we will first drive the reservoir, i.e. we will supply the training time series $\boldsymbol{X} = \boldsymbol{x}_{1:T} \in \mathbb{R}^{T\times N}$ to the ESN to generate a series of reservoir states $\boldsymbol{R} = \boldsymbol{r}_{1:T} \in \mathbb{R}^{T\times M}$. We then use Ridge Regression with regularization parameter λ and cost/loss function

$$L_{RR} = \|Y - \boldsymbol{R}\boldsymbol{W}_{out}^T\|_F^2 + \lambda\|\boldsymbol{W}_{out}\|_F^2, \tag{II}$$

where $\boldsymbol{Y} = \boldsymbol{x}_{2:T+1}$ are the regression targets, to compute $\boldsymbol{W}_{out}$ with closed-form solution

$$\boldsymbol{W}_{out} = \boldsymbol{Y}^T\boldsymbol{R}\left(\boldsymbol{R}^T\boldsymbol{R} + \lambda\boldsymbol{I}\right)^{-1}. \tag{III}$$

To generate from the ESN after training, we simply use eqs. (I) and after an initial warm-up phase of driving the system with data, we feed the reservoir its own predictions (i.e. we replace $\boldsymbol{x}_t$ by $\hat{\boldsymbol{x}}_{t-1}$).

You'll find a basic outline of an ESN implementation in the file '`sheet11_template.ipynb`'. The file "`lorenz_data.npy`" contains a Lorenz dataset.

TASKS

1. In the template code, implement all necessary functions to train and generate from an ESN. To this end, implement the functions and snippets that say "`your code here`". [50 pts]

2. Train and fit the ESN with the specified hyperparameters. Generate a trajectory from the model and plot 3D state space. If your ESN implementation is correct, you should get a decent fit of the dynamics. [25 pts]

3. In the code template, you are provided a measure that compares the power spectra of generated vs. ground truth trajectory using the Hellinger distance

$$D_H(\tilde{F}, \tilde{G}) = \sqrt{1 - \sum_{k=1}^{K} \sqrt{\tilde{F}(\omega_k)\tilde{G}(\omega_k)}}$$

where $\tilde{F}$ and $\tilde{G}$ are normalized power spectra of the respective trajectories. Perform a line search over different λ values, e.g. $\lambda = [0, 10^{-5}, 10^{-4}, \ \dots \ 10^{-2}, 1]$ and for every setting, plot the training loss and the Hellinger distance measure against λ. Do the minima of the curves align? Plot the generated attractors of both the model with lowest training loss and lowest Hellinger distance side-by-side. Which model has captured the attractor best? [25 pts]