Dynamical Systems Theory in Machine Learning & Data Science

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

Exercise 10

To be uploaded before the exercise group on January 15, 2025

1 Sparse Teacher Forcing

Consider the following autonomous RNN:

$$\boldsymbol{z}_t = \boldsymbol{F}_\theta(\boldsymbol{z}_{t-1}) = \boldsymbol{A} \odot \boldsymbol{z}_{t-1} + \boldsymbol{W}_1 ReLU\left(\boldsymbol{W}_2 \boldsymbol{z}_{t-1} + \boldsymbol{h}_2\right) + \boldsymbol{h}_1 \tag{I}$$

$$\boldsymbol{y}_t = \mathcal{I} \boldsymbol{z}_t, \tag{II}$$

where

$$\mathcal{I}_{kl} = \begin{cases} 1, & \text{if } k = l \text{ and } k \leq N \\ 0, & \text{else} \end{cases} . \tag{III}$$

is fixed and simply 'reads out' the first N hidden units of the RNN (read-out units). The $\odot$ operator denotes elementwise multiplication. Trainable RNN parameters are $\theta = \{\boldsymbol{A} \in \mathbb{R}^M, \boldsymbol{W}_1 \in \mathbb{R}^{M \times L}, \boldsymbol{W}_2 \in \mathbb{R}^{L \times M}, \boldsymbol{h}_1 \in \mathbb{R}^M, \boldsymbol{h}_2 \in \mathbb{R}^L\}$ where M is the latent dimension or state space dimension of the RNN, L is the size of an additional hidden layer and N the observations space dimensionality or number of observed dynamical variables. The nonlinearity is given by $ReLU(\bullet) = max(0, \bullet)$. This RNN is implemented in the `CustomRNN` class.

In this exercise, we will again try to reconstruct the chaotic Lorenz-63 attractor. In essence, training is similar to the previous exercise, in which we sample small batches from a Lorenz system trajectory with forcing signals and target values shifted by one time step, i.e. $\boldsymbol{y}_{t:t+\tau} = \boldsymbol{x}_{t+1:t+\tau+1}$. However, we use a specific observation model as well as another way of teacher forcing, called *sparse teacher forcing* (STF). This training routine tackles exploding gradients by directly forcing the hidden state of the RNN and hence manipulating the Jacobians involved in the BPTT algorithm. Here, we force the first N read-out neurons of our model using the ground truth data from the current training batch. That is, we directly replace latent state entries $z_{t',k}$ with observations $x_{t',k}$, $k \leq N$, producing the forced state $\tilde{\boldsymbol{z}}_{t'} = [x_{t',1} \dots x_{t',N}, z_{t',N+1} \dots z_{t',M}]$. This is done sparsely at time points $t' \in \mathcal{T} = \{n\tau_f \mid n\tau_f < T_{seq} \ \wedge \ n \in \mathbb{N}_0\}$, with forcing interval τ_f and sequence length τ. The latent state update is then given by

$$\boldsymbol{z}_{t+1} = \begin{cases} \boldsymbol{F}_\theta(\tilde{\boldsymbol{z}}_t) & \text{if } t \in \mathcal{T} \\ \boldsymbol{F}_\theta(\boldsymbol{z}_t) & \text{else} \end{cases} . \tag{IV}$$

The hyperparameter τ_f controls a trade-off between minimizing the exploding gradient problem and over-smoothing the loss landscape for low values (e.g. $\tau_f = 1$ in the extreme limit), as well as allowing the model to predict freely for a couple of steps and hence deal with errors in its

own dynamics unrolled in time.

We again minimize the MSE between observations and predictions

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}_{t:t+\tau}, \hat{\boldsymbol{y}}_{t:t+\tau}) = \frac{1}{\tau N} \sum_{t'=t}^{t+\tau} \|\boldsymbol{y}_{t'} - \hat{\boldsymbol{y}}_{t'}\|_2^2 \tag{V}$$

using stochastic gradient descent (SGD) and Backpropagation through time (BPTT).

The template code is given by the file `sheet10_template.ipynb`. The data of the Lorenz-63 system is contained in `lorenz_data.pt`. To make things a bit more challenging, the data is a delay embedding of the original attractor, contaminated with Gaussian noise.

TASKS

1. In the template, add code to perform STF. To this end, you need to add your own implementation at points in the code marked with comments `"your code here"`.

2. You can use the predefined hyperparameters to train the model on the data. Check for the quality of the reconstruction visually by generating a long trajectory of your trained model and visualizing the generated trajectories in observation space. To this end, choose an initial condition on the attractor, and initialize the remaining latent states of the RNN with zeros. Do the RNN dynamics follow those of the Lorenz system?

   Hint: You can play around with the hyperparameters, but you should already get decent results using the given defaults, assuming that you implemented the training routine correctly.

2 Maximum Lyapunov Exponent of an RNN

To check whether our RNN model has captured the dynamics correctly in exercise 1, we want to quantitatively measure reconstruction quality. To this end, we can use invariant properties of the underlying system, such as the maximum Lyapunov exponent. The maximum Lyapunov exponent of any map for some orbit $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T\}$ is defined as

$$\lambda_{max} := \lim_{T\to\infty} \frac{1}{T} \log \left\| \prod_{r=0}^{T-2} \boldsymbol{J}_{T-r} \right\|_2 \tag{VI}$$

where $\boldsymbol{J} = \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{z}_{t-1}}$ is the model Jacobian and $||\bullet||_2$ the spectral norm. The ground truth exponent is given by $\lambda^*_{max} = 0.906$.

If you did not manage to implement the training routine in exercise 1 of this sheet, load the pretrained model provided by `pretrained_rnn.pt`. To load the model, uncomment the corresponding line in the template notebook.

1. Argue what will happen if eq. (VI) is implemented naively and evaluated on a *chaotic* orbit.

2. Implement eq. (VI) and verify your theory by computing the Jacobian product along a chaotic orbit of your model trained on the Lorenz attractor. Store intermediate Jacobian norms, i.e. $\{\|\boldsymbol{J}_T\|_2, \|\boldsymbol{J}_T\boldsymbol{J}_{T-1}\|_2, \|\boldsymbol{J}_T\boldsymbol{J}_{T-1}\boldsymbol{J}_{T-2}\|_2 \ldots, \|\boldsymbol{J}_T\boldsymbol{J}_{T-1}\cdots\boldsymbol{J}_3\|_2, \|\boldsymbol{J}_{T-1}\cdots\boldsymbol{J}_3\boldsymbol{J}_2\|_2\}$, and plot them against the length of the product. What do you observe?

Hint: To compute the Jacobian, use the `torch.autograd.functional.jacobian` function.

3. The code template provides a numerically stable implementation to compute the maximum Lyapunov exponent, which re-orthogonalizes the Jacobian product regularly. Compute the maximum Lyapunov exponent using the provided function and compare the value to the ground truth. How large is the relative absolute error?

   Hint: Since the Lorenz data was generated with a read-out interval of $\Delta t = 0.01$, you will need to divide the model Lyapunov exponent by that value.

3 [BONUS] Reconstruction from a single variable

Can you manage to reconstruct the attractor dynamics from a single variable of the Lorenz system without explicit time delay embedding? To this end, train a model only on the x-component (first dynamical variable of the dataset `lorenz_data.pt`), i.e. $N = 1$. To examine your reconstruction, again generate a long trajectory using your RNN, plot the first component of the latent units and calculate the maximum Lyapunov exponent of the system.

Hint: It might be useful to include a warm-up of the hidden state: Choose a reasonable warm-up time T_W during which the RNN is forced every time step. After the warm-up, proceed to generate predictions with STF as in exercise 1. Exclude the generated predictions of the warm-up phase from the loss computation.