Dynamical Systems Theory in Machine Learning & Data Science

Lecturers: Daniel Durstewitz
Tutors: Manuel Brenner, Janik Fechtelpeter, Alena Brändle, Marc Pritsch, Lukas Eisenmann, Florian Hess, Max Ingo Thurm
WS2023/24

Final Project

To be uploaded before or on March 6th, 2024

For organizational purposes, please include your matriculation numbers when handing in the project. Please tell us if you need a grade, and if you handed in the exercise sheets as a group please check if you received a score on all the sheets. Lastly, also let us know if you need your grade for the lecture before the 6th of March.

The aim of the final project is to make use of the concept and ideas you have learnt in the last couple of months in a practical context by training a model on a time series, using it to generate a new time series, and assessing the quality of the reconstruction by a power spectrum distance.

As we have seen during the lecture, there are many different approaches that deal with dynamical systems reconstruction (some of them we will discuss in the final lecture.)
For the final project, you are asked to select a method from the literature where code is provided by the authors, and use it to reconstruct two time series derived from benchmark dynamical systems. To make things a little easier, we have compiled a list of suggestions. In case you have problems with the selection or are stuck with one of the approaches, feel free to reach out to us.

- Neural ODEs (e.g.Chen et al.). For this approach, you can e.g. implement a Neural ODE from scratch, using Pytorch package torchdiffeq package. In that case, please summarize the Neural ODE paper from Chen et al..
- Transformers (e.g. Wu et al.). Transformers have recently been very successful for large language models and computer vision, but have also been popular for time series forecasting. A tutorial for implementing this model can be found here.
- Reservoir Computing (e.g. Pathak et al.).
- Next Generation Reservoir Computing (Gauthier et al.). We've had mixed success with this model but you are free to try it and point out potential downsides.

Task 1. Write a short summary

Familiarize yourself with the content and theoretical background of the paper you selected and write up a short summary (1-2 pages) of its content in Latex. Do not use any LLMs, write it in your own words!

Task 2. Implement a performance measure

On moodle, we provide a code snippet called *psd.py*. This computes a power-spectrum distance between two input time series. It features a hyperparameter σ, which smoothes the spectra with a Gaussian kernel with width σ. Integrate this measure into your code by implementing a routine which calls the model, draws a random initial condition and generates a time series of length T, where T is the length of the test set. Then use this freely generated time series to compare the power spectra between the ground truth time series and the generated time series.

Task 3. Testing

On moodle, we provide you with two datasets. One is generated from the Lorenz-63 system that you have already encountered frequently in the tutorials, and the other from the Lorenz-96 model. For both time series, we added observation noise with 5 percent of the data variance to the train data make the reconstruction task more challenging.

Train the model. If reconstructions are not successful, play around with the respective hyperparameters of the algorithm (depending on the approach, it is of course not guaranteed that reconstructions will be successful in the end, but try not to give up too soon). Test the quality of reconstruction on the test data by computing the power spectrum distance between ground truth and reconstructed time series. Explore which values for the power spectrum smoothing factor which make sense for the given datasets.

Bonus exercise

Implement a routine that trains 20 models per dataset, and compute the average power spectrum distance across models. What do you observe?

Good luck!