Dynamical Systems Theory in Machine Learning & Data Science

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

Exercise 8

To be uploaded before the exercise group on December 18, 2024

Task 1.1 Reconstructing a dynamical system from data using PySINDy

In the lecture we discussed SINDy (Sparse Identification for Nonlinear Dynamical Systems), an algorithm developed by Brunton et al. that aims to uncover ODEs from time series data. The algorithm was first introduced in the paper "Discovering governing equations from data by sparse identification of nonlinear dynamical systems", which can be found under `https://www.pnas.org/content/113/15/3932`. There is also a useful introductory video on Steve Brunton's Youtube channel.
In this exercise we will make use of SINDy to reconstruct the Lorenz system. PySINDy is a python toolkit that implements SINDy, and offers useful functionalities that facilitate training the model, selecting libraries, and much more. You can install PySINDy using pip or conda. Generate a trajectory with 3000 time steps from the Lorenz system, with the parameters introduced on the last exercise sheet.

Task 1.2 Fitting a model

Start by learning a model from the time series, using the polynomial library up to order 6. Print the learned model. How does it compare to the ground truth ODE system? You can eyeball the reconstruction by comparing a 3D plot of ground truth data and generated data.

Task 1.3 Performance Measure

Estimating the quality of a reconstructed system is often challenging in practice, especially when the underlying equations are not known, and when high-dimensional data makes simple visualisations impossible. A commonly used performance measure for judging the quality of a reconstructed system is the forward prediction error. Implement an N-step prediction error that takes every point (minus the last N time points) of the ground truth time series (x_t) and iterates it forward via the reconstructed equations for N steps. The total MSE is the quadratic difference between all predicted points (x'_{t+N}) and the ground truth data (x_{t+N}), divided by the total number of time points ($T - N$). Compute this measure for different values of N. In case the analysis takes too long, check for the variance of the measure and see for how many time points it needs to be computed to achieve reasonably low variance. Based on what you know about the Lorenz system from the last exercise sheet and properties of chaotic systems, what would be a good value for N?

Task 1.4 Noise

Distort the time series by adding 10 percent observation noise on the ground truth data (that is, adding random noise drawn from a Gaussian ($\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, 0.1 \times \mathbf{I})$) for each data point independently. Rerun SINDy on the noisy dataset. Try several settings for the threshhold hyperparameter in the optimizer and see how the model changes. Search for an optimal setting of the threshhold by repeating your analysis of the prediction error. Further run your analysis for noise levels of 20 percent and 50 percent.

Task 1.5 Partial Observation
Assume we only have partial observation of the x coordinate of the Lorenz system (think of this e.g. as similar to only having measured a subset of neurons in the larger dynamical system that is the brain). Implement a time delayed embedding of the Lorenz system with sensible delay constant and embedding dimension, as motivated in the lecture. Fit a model on this dataset instead, and compute the forward prediction error again.

Task 1.6 Low Data Limit
Repeat your analysis, but only train your model on a subset of the original time series. What is the minimal number of time points you need to faithfully reconstruct the Lorenz system?

Task 1.7 Logistic Map
In the lecture we discussed the logistic map, defined by

$$x_{t+1} = r * x_t * (1 - x_t) \tag{I}$$

Implement the logistic map for a grid of 1000 values for r in the range of $(1.0, 4)$. Plot the bifurcation diagram of logistic map in the given range. Now generate a data set for each of the 1000 r-values and use SINDy to determine the r value form the generated data. Plot the true r-values against the estimated r-values an describe your finding.