Time Series Analysis & Recurrent Neural Networks

Lecturer: Daniel Durstewitz
Tutors: Georgia Koppe, Manuel Brenner, Daniel Kramer, Leonard Bereska
WS2020/2021

Exercise 09

To be uploaded before the exercise group on 3rd February, 2021.

In the last exercise, we captured a simple dynamical system (sinusoidal oscillation) with an RNN. Because the data were not noisy, we excluded the possibility of overfitting when encountering noise in the observations.

Task 1: Regularization
Regularization can be defined as *any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error* [1]. In the file *noisy_sinus.pt* you will find a sine wave overlayed with Gaussian noise. The goal is to capture the sine wave without the noise in our model (as in exercise 8):

$$z_t = \tanh(W_{xz}x_{t-1} + W_{zz}z_{t-1} + b_z) \tag{1}$$
$$x_t = W_{zx}z_t + b_x, \tag{2}$$

1. When optimizing the parameters of this network w.r.t. a mean squared error (MSE), why is this implicitly assuming a Gaussian noise distribution in the outputs $p(x_t|z_t) = \mathcal{N}(W_{zx}z_t + b_x, \mathrm{I})$?

2. Train the network with 10 hidden states for 400 epochs. Plot the resulting predictions. How does the overfitting manifest itself?

There are several ways to tackle overfitting:

3. **Reducing model capacity**: Reduce the number of hidden states until you capture a clean sine wave (without the model trying to reproduce the noise). Plot the predictions.

4. **Early stopping**: Reduce the number of training epochs, again, until the model does not overfit. Plot the predictions.

5. **Weight decay**: Regularize the model with a constraint on the magnitude of the weights in the loss: $L_w = L + \lambda \sum_{ij} w_{ij}^2$, where L is the loss and L_w the loss with weight decay.[2] Try different settings for the value of λ. What happens if regularized too strongly? Plot the predictions for different λ.

6. In which sense is weight decay equivalent to early stopping?

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] In PyTorch you can directly do this with an option in the optimizer: `tc.optim.Adam(model.parameters(), weight_decay=lambda)`.