Time Series Analysis & Recurrent Neural Networks

lecturer: Daniel Durstewitz
tutors: Georgia Koppe, Leonard Bereska
WS2019/2020

Exercise 9

To be uploaded before the exercise group on January 8th, 2019

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 modication 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):

$$z_t = \tanh(W_{xz}x_{t-1} + W_{zz}z_{t-1} + b_z) \quad \text{(I)}$$
$$x_t = W_{zx}z_t + b_x, \quad \text{(II)}$$

- When optimizing the parameters of this network w.r.t. a mean squared error (MSE), why is this implicitly assuming Gaussian noise added to the outputs x_t?
- 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:

1. **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.
2. **Early Stopping**: Reduce the number of training epochs, again until the model does not overfit. Plot the predictions.
3. **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. In PyTorch you can directly do this in the optimizer with the so-called weight decay:
`tc.optim.Adam(model.parameters(), weight_decay=lambda)`.
Try different settings for the value of λ. What happens if regularized too strongly? Plot the predictions for different λ.

- In which sense can weight decay and early stopping be seen as equivalent?

Christmas Bonus: PyTorch Tutorials
To bring everyone up to speed with PyTorch, the following tutorials are mandatory for those of you who still feel insecure about back-propagation in PyTorch.
`https://pytorch.org/tutorials/beginner/pytorch_with_examples.html#tensors`
`https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html#`
But feel free to have fun also with the other tutorials on the PyTorch website. Merry Christmas!

References

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