EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Computer Science Department
Chair of Cognitive Modeling

Winter semester 2019/2020

Advanced Neural Networks

Exercise sheet 03

Release: December 12, 2019 Deadline: January 12, 2020

General remarks:

- Download the file `exercisesheet03.zip` from the lecture site (ILIAS). This archive contains files (Python code and data), which are required for the exercises.
- Ideally, the exercises should be completed in teams of two students. Larger teams are not allowed.
- Add a brief documentation (pdf) to your submission. The documentation should contain protocols of your experiments, parameter choices, and discussions of the results.

Exercise 1 – RNNs Recapitulation [30 points]

(a) Even though that LSTMs (or other gated memory units) use smooth, continuous activation functions (particularly for gating), they can learn to count and solve discrete tasks that involve precise and sharp calculations. Explain why this is possible and what happens during training. [4 points]

(b) Consider a single LSTM unit with D memory cells (recall that an LSTM can have multiple CECs per unit). Following the backward pass equations (lecture 3, slide 13) the forget gate gradient can be calculated with

$$\delta_\phi^t = \varphi'_\phi(net_\phi^t) \sum_{c=1}^{D} \zeta_c^t s_c^{t-1}. \tag{1}$$

Derive this equation step by step from

$$\delta_\phi^t = \frac{\partial E}{\partial net_\phi^t}. \tag{2}$$

You can assume the following definition as given

$$\zeta_c^t =_{\text{def}} \frac{\partial E}{\partial s_c^t}. \tag{3}$$

Keep track of all non-trivial intermediate steps. [16 points]

(c) Consider a 3-dimensional multidirectional RNN (for volume segmentation) with 3 input neurons, 16 sigmoidal hidden units per hidden layer (one independent hidden layer per direction, following the standard formulation), and a softmax output layer with 20 classes. Assume that there are no biases. How many hidden layers does this networks contain and how many weights does it have in total? [6 points]

(d) Name one advantage and one disadvantage of MDMDRNNs compared to feed forward CNNs. [4 points]

Exercise 2 – Temporal Convolution [20 points]

(a) What are the main differences between a temporal convolutional ANN and a convolutional LSTM network? What prior inductive biases (data expectations) are implicitly generated by the design of a convolutional LSTM network? [6 points]

(b) Why may a convolutional LSTM be particularly suitable for weather prediction problems? [4 points]

(c) Calculate the number of weights a temporal convolutional ANN has that consists of three layers (as shown on slide 6 of the lecture slides) with $d = 1, 2, 4$ for the three layers, temporal filter size $k = 3$, data input size $n = 300$ and $m = 200, 100, 50$ convolutional kernels per layer. [10 points]

Exercise 3 – Predicting Spatio-Temporal Data with PyTorch [50 points]

In this exercise, two architectures are compared at predicting a spatio-temporal process, the circular wave, as seen in Figure 1. To model this process—by approximating the field activity at each pixel (see Figure 1, right)—the PyTorch library will be used.

(a) Setup [0 points]

Make sure you have Python 3 installed. Go to the PyTorch web page[1] to specify your system and install PyTorch. We strongly recommend to set up a virtual environment, using e.g. anaconda, to prevent messing up the global system. [0 points]

(b) Temporal Convolution [15 points]

In this exercise part, you can verify whether PyTorch is installed properly. Unzip the folder exercisesheet03.zip and find the directory **tconv_arch**. Run `main.py` to run the pre-trained model *test_model* and observe its performance.

A new model can be trained by defining a new *model_name* in line 14 of `main.py` and changing the *mode* (line 17) from 2 to 1. Model weights will be stored in the saved models folder and can be tested by setting the *model_name* and *mode* fields accordingly.

Calculate the closed loop error between network output and target over 65 time steps, by choosing a *seq_len* of 80 (training and validation files consist of 40 time steps, testing files of 80 time steps) and use 15 *teacher_forcing_steps* and 65 *closed_loop_steps*. The error calculation can be implemented around line 96 of the `tester.py` file.

Report the average error over ten sequences, including standard deviation.

[1] `https://pytorch.org/get-started/locally/`

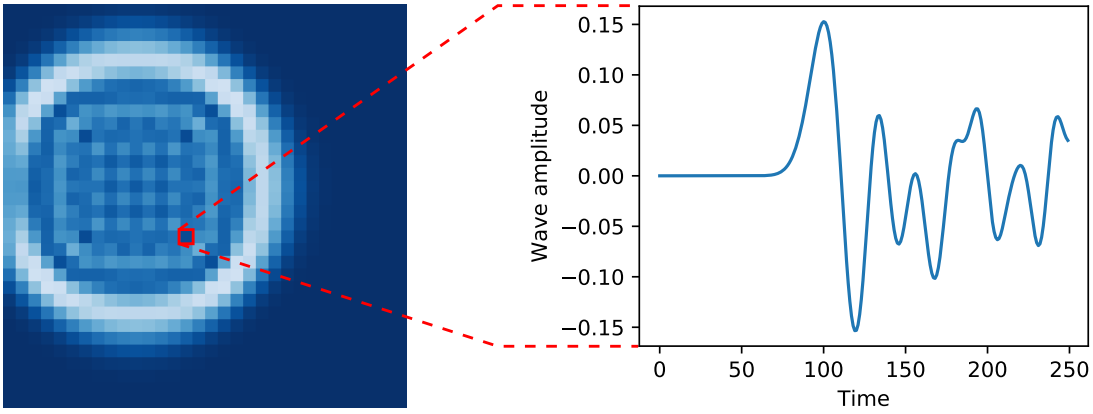

Figure 1: Left: circular wave process that is to be modeled. Right: field activity at one pixel.

(b) DISTANA [35 points]

The model implementation of a DISTributed Artificial neural network Architecture (DISTANA) that puts a weight sharing Prediction Kernel (PK) on each pixel of the two dimensional simulation domain is provided in folder **distana_arch**. PKs next to each other can exchange information, i.e. each PK has eight neighbors to spread information laterally. The given implementation is incomplete, meaning the PK does not consist of layers to appropriately model the wave.

Initially, train a model with the given PK, test it and report the closed loop error over 65 time steps between network output and target as above, that is compute the average over at least ten sequences and include a dispersion measure.

Adapt the PKs such that the networks models the wave properly. Changes should be made in both the `init` and the `forward` method of `prediction_kernel.py` in the **kernel_architecture** folder. You may also look at the files `kernel_net` (forward pass) and `kernel_variables` (tensor class) to realize deeper changes.

Optional (bonus): Compare models that do and do not use bias neurons, tested on a 40×40 grid (change the *pk_rows* and *pk_cols* variables in `kernel_variables.py` in the **kernel_architecture** folder to 40 each – only for testing!). Report your observations and provide an explanation.

For questions concerning this exercise, please refer to the forum or contact Matthias Karlbauer (**matthias.karlbauer@uni-tuebingen.de**).