Exercise 2: Backpropagation and Feedback Alignment

Upload the modified files `backprop_functions.py`, `feedback_alignment_functions.py` and a **typed** pdf file of your answers to moodle by noon on 14.10.2020.
Provide your name, student ID and email address as comment at the top of the sourcefiles. Take as name for your pdf file last_name_first_name_tutorial2.pdf and provide your name, student ID and email at the top of the file.

This tutorial contains two main parts. In the first part, you will have to implement the error-backpropagation (BP) algorithm. In the second part, you will have to implement the feedback alignment (FA) algorithm and perform some experiments for gaining a better understanding of FA. During the exercise session on 30.09.2020, we will discuss the first part on BP. During the exercise session on 07.10.2020 we will discuss the second part on FA. We encourage you to download already the source code before the start of the exercise session, such that you can ask questions during the exercise session if you are having troubles with setting up everything.

1 Part I: Backpropagation

Please download and unzip the corresponding source files, which contain all implementations, the test cases as well as the documentation.
You should start by **opening the documentation**. Therefore, you have to open the file `docs/index.html` in your webbrowser.
Furthermore, you have to **setup your Python environment**. You can do so manually (e.g., by installing all missing packages) or use the conda environment that we provide with the file `tutorial2_env.yml`. The provided conda environment can be installed via[1]

```
$ conda env create -f tutorial2_env.yml
$ conda activate tutorial2_env
```

The goal of this exercise is to implement the missing code in the `forward()` and `backward()` functions in module `lib/backprop_functions.py`.
The corresponding computational rules were derived in the tutorial session and are described in the documentation (that you opened in your webbrowser).
To be precise, the following methods have to be implemented:

- `lib.backprop_functions.LinearFunction.forward()`
- `lib.backprop_functions.LinearFunction.backward()`
- `lib.backprop_functions.SigmoidFunction.forward()`
- `lib.backprop_functions.SigmoidFunction.backward()`
- `lib.backprop_functions.MSELossFunction.forward()`

[1]See here for more details: `https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file`

- `lib.backprop_functions.MSELossFunction.backward()`

The documentation describes how you can automatically test your implementation. The instructors have a very similar withhold set of private test cases that are used to grade the exercise.

2 Part II: Feedback Alignment

The goal of this exercise is to implement feedback alignment, based on the coding exercise of last week and to gain some more insight in the workings of feedback alignment by investigating simple toy examples. You should complete the missing code in the `forward()` and `backward()` functions in module `lib/feedback_alignment_functions.py` and upload a pdf file with the answers to the exercises.

2.1 Implementation

The corresponding computational rules were showed in the tutorial sessions and are described in the documentation (that you opened in your webbrowser).
To be precise, the following methods have to be implemented:

- `lib.feedback_alignment_functions.LinearFunctionFA.forward()`
- `lib.feedback_alignment_functions.LinearFunctionFA.backward()`

The documentation describes how you can automatically test your implementation. The instructors have a very similar withhold set of private test cases that are used to grade the exercise. For this part of the exercise, you need to submit the modified `feedback_alignment_functions.py` file.

2.2 Experiments with feedback alignment

In order to gain a better understanding of feedback alignment, you will experiment with some simple toy examples. For answering the questions, make sure to check the feedback alignment paper [1] and the supplementary materials of the paper. The commandline code needed for running the experiments is given in the documentation (that you opened in your webbrowser).

Note. The questions marked with [**Optional**] will not be taken into account for grading the exercise and serve only to challenge you into some deeper thinking.

2.2.1 Linear case

Following the paper, we first investigate the case of a linear network (without sigmoid nonlinearities).

- Run the commandline code for the linear student-teacher regression that is provided.
- Save the plot of the angle between B and W^T and put them in your report.
- Based on these plots, can you explain why feedback alignment succeeds in diminishing the loss of the network? (Link FA back to the backpropagation equations).

- [**Optional**] As this is a linear network, the input-output relation of this network is a simple linear function. Does the network need to send useful feedback signals to the hidden layer in order to improve results, or can it learn everything worth learning (for a linear function) based on only its output layer? Tip: look at the dimensions of the network.

2.2.2 Nonlinear case

Now we investigate the nonlinear case of the student-teacher network (with sigmoid nonlinearities).

- Run the commandline code for the nonlinear student-teacher regression for a network with one hidden layer.
- Save the two plots of the angles between B, W^T and $W^\dagger$ and put them in your report.[2]
- Based on these plots, do you think that feedback alignment also works in the nonlinear case?

- [**Optional**] Can you explain why W aligns with B^T and $B^\dagger$ at the same time?

- Run the commandline code for the same nonlinear student-teacher regression but now with backpropagation.
- Compare the results with feedback alignment. Is there a difference in performance? If so, can you give one of the possible reasons?

2.2.3 Polynomial fitting

As a last part, we investigate the polynomial fitting of a one-dimensional input to a one-dimensional output. We will run the training for many iterations, to investigate to which solutions BP and FA converge.

- Run the commandline code for the polynomial fitting with feedback alignment.
- Save the plot of the polynomial regression and put it in your report
- Run the commandline code for the polynomial fitting with backpropagation.
- Save the plot of the polynomial regression and put it in your report
- Compare the results. Is there a difference in performance? What about overfitting? If so, can you explain why?

2.3 Further reading

If you are interested in feedback alignment and other similar methods, make sure to check the video of Hinton in Stanford `https://www.youtube.com/watch?v=VIRCybGgHts&t=3967s`

[2] $W^\dagger$ denotes the Moore-Penrose pseudo-inverse of W, which is a generalization of the inverse for matrices of arbitrary size.

References

[1] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, p. 13276, 2016.