Assignment 1:

Outline:

In the first assignment, we had to implement a simple neural network in python. A code stub was provided.

Following should be implemented:

- Possibility to use 2-4 layers
- Sigmoid/tanh and ReLU for hidden layer
- Softmax output layer
- Optimization via gradient descent and Stocastic gradient descent
- Gradient checking code
- Weight init with random noise

After implementing the neural network, we should train it, test the result and plot some data.

Result:

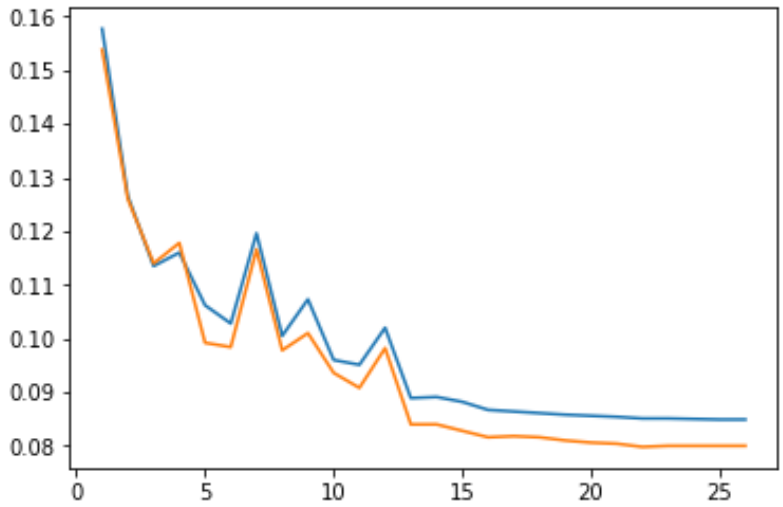
I got a lost in the SoftmaxOutput layer. I had a stupid bug, where I double computed the softmax -> so the loss was calculated wrong. I finally managed to get the network producing reasonable results. I accomplished all tasks except of the own gradient checking code. I had to save time I lost on the softmax bug and the cheat solution provided seemed to be good enough.

Results on the MNIST Data set:
I used 3 hidden Layers with following properties:

1. 22 Perceptrons with ReLU activation
2. 77 Perceptrons with ReLU activation
3. 10 Perceptrons with None (linear activation)

I used SDG with a learning rate of 1.2 and a (mini-)batch size of 15

I managed to get my validation error as low as 8% with above parameters (see learning curve (L1) below).



L1: Learning curve (training error: blue, validation error: yellow)