Deep learning Assignment 1 (2020)

MPhil in Computational Biology

November 26, 2020

If there are errors found, I will update the assignment. **Due date: 2020-12-19 23:45**

Please submit your report to moodle, anonymised as before. Your report must be a maximum of ten pages, excluding the appendix. Your appendix should contain only a copy of your code.

This assignment is worth 50% of your overall mark for this module.

1 Perceptron [10 marks]

(Adapted from Dayan and Abbott's Theoretical Neuroscience text.)

Construct a perceptron that classifies 10 binary inputs (each input ± 1) according to whether their sum $\sum x_i$ is positive or negative. Show how the network error changes over epochs. How well does your network generalise? Describe how you created your training and test set.

Repeat this training protocol, but this time attempt to make the output of the perceptron classify according to the parity of the inputs, which is the sign of their product $\prod x_i$. Why is this example so much harder than the first case?

You may adapt the code example `per.R` in class for this if you wish.

2 Self-supervised learning [20 marks]

Write your own multi-layer perceptron with the following specification: eight input units, three hidden units (sigmoidal activation) and eight output units (sigmoidal activation). There are eight 8-d input vectors $\mathbf{x}$, for each of the eight cases where one input unit is "1" and the other seven units are "0". Your task is to solve the self-supervision problem of reproducing the input vector on the output vector, so $\mathbf{t} = \mathbf{x}$.

If your network can solve the task, how does it solve the problem? Does including a momentum term change learning?

You may adapt the code `xor.R` if you wish.

3 MNIST classification [20 marks]

Describe how various hyper-parameters (number of layers, number of units, learning rate, choice of optimisation algorithm) affects the classification performance of MNIST. I suggest you start from the `mnist_bp.Rmd` example, but you are free to use other methods if you wish. What choices of hyper-parameters give you the best performance? Are there any digits that are particularly difficult to classify?