

Introduction to Programming with Scientific Applications (Spring 2020)

Final project

Study ID	Name	% contributed
201706722	Mikkel Werling	50
201707639	Victor Møller Poulsen	50

max 3 students

All contributed equally. Worked as a group. #SmellsLikeTeamSpirit

Note on plagiarism

Since the evaluation of the project report and code will be part of the final grade in the course, **plagiarism in your project handin will be considered cheating at the exam.** Whenever adopting code or text from elsewhere you must state this and give a reference/link to your source. It is perfectly fine to search information and adopt partial solutions from the internet – actually, this is encouraged – but always state your source in your handin. Also discussing your problems with your project with other students is perfectly fine, but remember each group should handin their own solution. If you are in doubt if your solution will be very similar to another group because you discussed the details, please put a remark that you have discussed your solution with other groups.

For more Aarhus University information on plagiarism, please visit

My footer

<http://library.au.dk/en/students/plagiarism/>

Learn or let die

1. Design of the structure of the code

We partitioned our code into six files. "read_write_helper.py", "math_helper.py", "plots_helper.py", "network_helper.py", "tests.py", "learn.py".

The "read_write_helper.py", "math_helper.py", "plots_helper.py" and "network_helper.py" files reflect our attempt at partitioning the tasks into coherent blocks. The idea is that each file can be loaded as a module and the functions can be used in other documents (e.g. to train a neural network in "learn.py").

"read_write_helper.py" handles tasks B, C, F and G. These functions in this module all have to do with reading and writing files.

"math_helper.py" handles tasks H, I, J, K, L and M. It contains functions which deal with linear algebra and math. It basically contains functions that one would usually turn to the numpy library for. Doctests were run on the functions in this module as the mathematical nature of these functions lends itself very nicely to these. All subsequent files rely on this module.

"plots_helper.py" handles tasks D, N and O, which are the plotting related tasks. It contains functions which plot images of digits as well as visualize the weights of a neural network as heatplots.

"network_helper.py" handles tasks P, Q, R and more. This module contains functions more directly related to updating, training and testing neural networks. Notably, the function which completes task R is in this doc-

ument, and this is arguably the culmination of the project (NB: there might be optionals here).

"test.py" does not handle any specific tasks. It tests the assumptions of the functions from previous modules as well as showcases functionality. Because we wanted to test that assumptions (i.e. assertions) worked as intended this document (intentionally) returns several errors (mainly ValueError) when we specify invalid input to our functions. It also serves to showcase how the plotting functions work, as it was less obvious to us how to test these with docstrings.

"learn.py" does not handle any specific tasks either. It uses the functions of previous modules to train an initially random neural network on the training dataset from MNIST. Afterwards this network is validated against the test dataset from MNIST (NB: this was not actually a task, but just seemed reasonable).