M3C 2018 Homework 2

Due date/time: 16/11/2018, 23:59 GMT

Clarification

11/11/18: The definition of the testing error in the online neural network notes had a typo, the notes have been corrected, and the error should be $1-(N_{correct}/T)$ rather than $(1-N_{correct})/T$.

Getting Started: If needed, update your local copy of the course repo (*sync* and *pull*). The homework/hw2/ directory contains the files, *hw2_template.py*, *hw2_template.f90*, *hw2_main.f90* and *data.tgz*. Make a new directory outside of the course repo, and place *dev* copies of the template files in this new directory along with the the 'main' and data files. Ultimately, the final files that you submit should be titled *hw2.py* and *hw2.f90*. You will have to 'unpack' the data at the Unix terminal using:

```
$ tar zxf data.tgz
```

This should create a large (122 MB) file named *data.csv* which contains image data for the handwritten numerical digits that were used in lab 3. Both *hw2_template.py* and *hw2_main.f90* contain code for reading this file.

Browse through the two template files. They contain a number of functions/subroutines, and you will have to add code as described below. First, however, you should *add your name and 8-digit college id to the docstring/comment at the top of each dev file.*

Note: As you work through the tasks below, you may add module variables and additional subroutines and functions as needed. Do not remove module variables or modify input/output of existing routines without first asking for permission (unless the code documentation indicates otherwise).

In this assignment you will implement and analyze two "neural" classification methods which will be used to classify images of handwritten digits as even (*y*=0) or odd (*y*=*1*). The two methods to be investigated are: 1) the single neuron model (SNM) and 2) a neural network model (NNM) with one internal layer and a single neuron in its output layer. Notes describing these models have been posted online and should be read carefully before proceeding with this assignment.

Part 1: Single Neuron Model

1) (20 pts) Complete the subroutine *snmodel* in *hw2_dev.f90* so that it computes the cost function, *c*, and its gradient, ∇c, for the SNM with fitting parameters (the weight vector and bias) provided as input to the subroutine. The least-squares cost described in the NN notes should be used. Note that the input data and labels should be set elsewhere before the subroutine is called.

2) (15 pts) Complete the function *snm_test* in *hw2_dev.py* so that it (a) trains the SNM by finding the fitting parameters that minimize the cost function computed in *snmodel*, and (b) computes the testing error after the model has been trained and the optimal fitting parameters have been found. The fitting parameters obtained after training and the testing error should both be returned by the function. The input parameter, *omethod* should set the optimizer that is used. When *omethod=1*, *l-bfgs-b* should be used to minimize the cost, and when *omethod=2*, the SGD routine included in the *nmodel* module should be used. In both cases, the cost and gradient should be evaluated using *snmodel*. The initial fitting parameters should be sampled from a random normal distribution. The testing data throughout this assignment should consist of the final 1e4 images contained in the provided dataset.

3) (5 pts) In the docstring for *snm_test*, discuss the expected advantages and/or disadvantages of training and testing the single neuron model purely in Python vs. the Python+Fortran approach used here. Pick 2 key points and provide clear concise explanations.

Part 2: Neural Network Model

1) (30 pts) Complete the subroutine *nnmodel* in *hw2_dev.f90* so that it computes the cost function, *c*, and its gradient, ∇c. for the NNM with fitting parameters (the weight matrix and bias vector) provided as input to the subroutine. The least-squares cost described in the NN notes should again be used. The number of neurons in the inner layer is set by the input parameter, *m*, and as noted above, there should be one output neuron. As with *snmodel*, the input data and labels should be set elsewhere before the subroutine is called.

2) (5 pts) Complete the function *nnm_test* in *hw2_dev.py* so that it (a) trains the NNM by finding the fitting parameters that minimize the cost function computed in *nnmodel*, and (b) computes the testing error after the model has been trained and the optimal fitting parameters have been found. The fitting parameters obtained after training and the testing error should both be returned by the function. The input parameter, *omethod* should set the optimizer that is used. When *omethod=1*, *l-bfgs-b* should be used to minimize the cost, and when *omethod=2*, the SGD routine included in the *nmodel* module should be used. In both cases, the cost and gradient should be evaluated using *nnmodel*. The initial fitting parameters should be sampled from a random normal distribution. The testing data again should consist of the final 1e4 images contained in the provided dataset and the corresponding labels.

3) (25 pts) Analyze and compare the performance of the single neuron and neural network model training codes as well as the performance of the models themselves when applied to the even/odd classification problem. You should keep the test data fixed to 10000 images and it is sufficient to keep the SGD learning rate set to $\alpha = 0.1$. Otherwise, it is up to you how to vary parameters and investigate this problem. Add python code to the function *nm_analyze* in *hw2_dev.py* which produces 2-4 figures which illustrate the most important qualitative trends you identify. The docstring of *nm_analyze* should contain a clear, concise discussion of these trends and your main conclusions. Save your figures as *.png* files with the names *hw21.png*, *hw22.png*, …, and submit them with your codes.

Notes: 1. All figures created by your code should be well-made and properly labeled. The title of each figure should include your name and the name of the function which created it. There should be one plot per figure.

2. Marking will primarily focus on the functionality of your code – does it produce the right answer(s) and function as required – and the soundness of your analysis. However, points may be deducted if there are substantial inefficiencies in the code,

3. You do not need to average over several realizations of training and testing calculations even though random numbers are used in a few places.

4. For part 2, question 3, you may use MLPClassifier in scikit-learn in place of your implementation of the NNM, however, the maximum possible score for the question will then be 20 points.

5. It is fine if your final version of *nm_analyze* requires several minutes, but you should consider initially developing your code with modest problem sizes (e.g. avoid using all 60000 training images).

Submitting the assignment

To submit the assignment for assessment, go to the course blackboard page, and click on the “Assignments” link on the left-hand side of the page, and then click on Homework 2. Click on “Write Submission” and add the statement: “This is all my own unaided work unless stated otherwise.” If you are a CDT student, add a 2nd line with “CDT Fluids”, or “CDT TSM” as appropriate.

Click on “Browse My Computer” and upload your final files, *hw2.f90*, *hw2.py*, *hw21.png*, *hw22.png*, … Finally, click “Submit”.