COM3013 Coursework – preliminary, subject to external examiner approval

Module Name: Computational Intelligence

Module Code: COM3013

Convenor: Frank Guerin

Coursework due: 4:00pm, Monday, 5 January 2022

(**Your coursework report must be uploaded on SurreyLearn.**)

Maximum mark: **50 (50% of the whole module)**

Note: Academic Misconduct: Coursework will be routinely checked for academic misconduct. Your submission must be your own work. Please read the Student Handbook to ensure that you know what this means.

All functions in the DEAP library or PyTorch / NumPy / SymPy are allowed to be used for this coursework, and any other libraries within reason (e.g. for visualisation).

Please include your full name, URN and email address in the report. You may use any formatting you like for the report. You must include everything in one PDF file, and make it clear (with headings) where each part of a question starts. *Your report should be single-spaced.*

Submit a single PDF file. *The code you used to produce the results should be in the report. i.e. everything should be pasted into one document, do not submit code as separate files.*

The purpose of the coursework is to enable you to gain hands-on programming skills for using evolutionary algorithms to solve optimisation and learning problems.

Some tips will be posted in Surrey Learn (under Course Materials, Coursework 2) to help with difficult parts of code.

Please explain in detail how your results are obtained!

1. The main objective of this task is to train feed-forward multi-layer perceptron networks with two hidden layers to approximate the following function:

$$y = \sin(3.5\,x_1 + 1.0)\cos(5.5\,x_2), \qquad x_1, x_2 \in [-1, 1]$$

Implement the algorithm in a computer language of your choice, although you are strongly recommended to use the Python libraries used in class.

1.1 Visualise the function with a 3D surface plot, in the given range. Use of NumPy for numbers is recommended. [**3 marks**]

1.2 Randomly generate 1100 samples for x_1 and x_2 within [-1,1]. Calculate the corresponding y values for the 1100 samples. Use 1000 of them as the training dataset, and the other 100 samples as a test dataset. It is recommended to store your data as a tensor (to make it easier to use in the neural net later), e.g.

```
x = torch.as_tensor(data, dtype=torch.double)
```

Visualise the training and test data (two separate plots) in a three-dimensional graphics,
e.g. scatter3D(x1, x2, z, c=z). [**3 marks**]

1.3 Assume that the neural network has six hidden neurons in each of the hidden layers and the network is fully connected. There is a threshold/bias connection for all hidden nodes and the output node. The activation function used is a sigmoid function in the hidden neurons, and a linear activation function in the output neuron. Write the code that creates the network [**3 marks**]

1.4 Write the following two functions:

weightsOutofNetwork: extracts all the weights of the network and puts them in one list; returns this list.
weightsIntoNetwork: takes as input a list of all weights of the network and uses them to set the weights of the network

Write a test to check if the weights are retrieved and inserted in the network correctly. Show the following test results in your report: use weightsOutofNetwork to retrieve all weights; print out the first layer of weights (connecting input to hidden); change three of these weights; insert all weights back in the network using weightsIntoNetwork; use weightsOutofNetwork to retrieve all weights again, and print out the first layer to check. Highlight the changed weights, e.g. in bold. [**7 marks**]

1.5 Use a binary coded genetic algorithm (with Gray coding) for optimising the weights of the neural network to fit the function, by minimising the mean squared error on the training dataset. Your evaluation function should extract weights from the chromosome and insert all weights in the network using weightsIntoNetwork, the loss of the network on the training data should then be used as the fitness of the individual.

Use 30 bits for encoding each weight and limit weights to the range [-20,20]. It will help to initialise your individuals to a smaller range (e.g. [-1,1]).
Experiment with hyperparameters to get a good result. Show your complete code, with hyperparameters clearly named at the top. Give a brief justification for your choice of hyperparameters.
Show a plot of the training and test error across the generations. [**10 marks**]

1.6 Show a 3D surface plot of the function implemented by the neural network across the range [-1,1]. To do this compute the network's output for a grid of values uniformly covering the range. i.e. similar to 1.1 except using network output instead of the mathematical function. [**2 marks**]

1.7 Write a function which takes a list of weights (output from weightsOutofNetwork) and returns a chromosome (i.e. a long list of binary bits, in Gray coding). The function should check if weights need to be pushed back into the range [-20,20]. Write a test to test your function: test if weights can be put into a chromosome, and retrieved, and turned back into weights. Show the results of your test for a small number of weights. Note that numbers retrieved will not be exactly the same as those input because of a loss of accuracy in conversion. [**4 marks**]

1.8 Embed the ***Rprop learning*** in the genetic algorithm as a local search method (lifetime learning). Use the ***Lamarckian learning approach***, i.e., the weight changes in the lifetime learning are encoded back to the genotype. Implement 30 iterations of local search in each generation. Implement the above memetic algorithm by extending the code above. Explain hyperparameter choices. Show a plot of the training and test error across the generations, and also a 3D surface plot of the function implemented by the neural network across the range [-1,1]. [**9 marks**]

1.9 Implement the ***Baldwinian learning approach*** to replace the Lamarckian approach in Question 1.8 and plot the results. Show a plot of the training and test error across the generations, and also a 3D surface plot of the function implemented by the neural network across the range [-1,1]. [6 marks]. Analyse the difference in the results compared to those obtained in Question 1.8 and discuss possible reasons [3 marks]. [**9 marks**]