EEL-5813 Neural Networks SUMMER 'C' 2019

INDIVIDUAL PROJECT II **Backpropagation Learning (Basic)**

Due: ____________________

I. Objective:

In this project the student will implement the Backpropagation algorithm for a Multi Layer Perceptron (MLP), towards the solution of a pattern classification problem. The student will experiment with the original Backpropagation algorithm,

NOTE: USE THE SAME TRAINING AND TEST PATTERNS AS FOR PROJECT I (Except that the TARGETS will now consist of 5 bipolar values, as explained below).

I. MULTILAYER NETWORK WITH BACKPROPAGATION

Design, train and test a 2-layer (i.e., 1 "hidden layer"), 5-output (a^2_1, a^2_2, a^2_3, a^2_4, a^2_5) network, for the association of the 5 classes in the training set ("A", "E","I","O","U") to the groups of 5 target values, as follows:

Target component	For "**A** s" (p1 to p5)	For "**E** s" (p6 to p10)	For "**I** s" (p11 to p15)	For "**O** s" (p16 to p20)	For "**U** s" (p21 to p25)
t_1	**+1**	-1	-1	-1	-1
t_2	-1	**+1**	-1	-1	-1
t_3	-1	-1	**+1**	-1	-1
t_4	-1	-1	-1	**+1**	-1
t_5	-1	-1	-1	-1	**+1**

TRAIN THE NETWORK WITH STANDARD BACKPROPAGATION

You will need to experiment with the following parameters:
- Number of units in the first ("hidden") layer (i.e., the number of a^1 activations).
- Adaptation rate, α (In this assignment you will use a single, constant value of α for the whole network).
- Stopping criterion (i.e., condition that will stop the training when fulfilled.)

For this assignment; initialize all the weights and the biases with "small random numbers". So, for example, if you are using 10 processing elements in the hidden layer, you could use these Matlab statements to initialize weights and biases:

```
W1 = 0.25 *  randn(10,20); W2 = 0.25 * randn(5,10);
b1 = 0.25 * randn(10,1); b2 = 0.25 * randn(5,1);
```

Monitor the training process plotting:
- The mean squared error for each epoch
- The evolution of any 5 of the weights or biases (at the end of each epoch).

INCLUDE THOSE PLOTS IN YOUR REPORT FOR ONLY 3 TRAINING CASES: "Worst", "Intermediate" and "Best". In each case indicate training parameters, stopping criterion and list final weight values.

Test the "Best" set of trained weights (ONLY), applying the test sets.
Here, to make a direct comparison to the bipolar 5-value target vector, you will need to threshold the outputs of the network (for example applying hardlims()) and only consider a "hit" if all 5 values of the target are matched.

Indicate the "hits" and "misses" for each pattern in each of the three (TSETI, TYSETII, TSETIII) test sets, as well as the "hit ratio".

Comment on what aspects made different sets of training conditions better or worse, including your interpretation as to why that happened.

II. "Fault Tolerance"

II.1 Destroy (make zero) about 20 % of the trained weights of the multilayer network that obtained the "best" performance with the test patterns. The selection of which weights to destroy is arbitrary, but the number of weights destroyed in a layer should be proportional to the number of total weights in that layer (For example, if there are 30 weights in the hidden layer and the total number of weights is 50, then, approximately 60% of the weights destroyed should be weights in the hidden layer). Then test the network with the three test sets and evaluate again hits, misses and hit ratio.

II.2 Destroy an additional 20% of the weights (distribution also proportional to the total number of weight in each layer) and test the network, evaluating its performance.

Comment on your observations.

REPORTING:

You must turn in an individual, self-contained **printed** report, covering all the aspects requested above. In addition, by the same deadline, you must upload all your code, as a single .ZIP file, to Canvas. (The corresponding Canvas Assignments will be created at least 3 days before the deadline)

- *PLEASE INCLUDE LISTINGS OF ALL YOUR SOURCE CODE IN YOUR PRINTED REPORT.*
- *You must PROGRAM BACKPROPAGATION YOURSELF. You CANNOT used the pre-programmed implementations available in Matlab's Toolboxes (e.g. Neural Nets).*