Homework No. 2
Due Friday, March 4, 2022, noon

Student certification:

Team member 1:

Print Name: _____ *Date:* _____
I have contributed by doing the following: _____
Signed: _____ *(you can sign/scan or use e-signature)*

Team member 2:

Print Name: _____ *Date:* _____
I have contributed by doing the following: _____
Signed: _____ *(you can sign/scan or use e-signature)*

Team member 3:

Print Name: _____ *Date:* _____
I have contributed by doing the following: _____
Signed: _____ *(you can sign/scan or use e-signature)*

2.1 Python, hard/soft activation function, delta rule (5pts)

2.1.0 Download two Python programs (0 pts).
Download two Python programs *perceptron_hard.py* and *perceptron_soft.py* (save these scripts by the name listed in the header of script). Run them, inspect their output files. Try to understand the functionality of these programs. You will use these programs as skeletons for the following sections of this homework.
Report: None.

2.1.1 Write a program in Python (based on 2.1.0) to train a neuron implementing the truth table from HW1.4 (Homework 1, problem 4). Use a hard activation function and perceptron learning rule. For initial weight set use (1, 1, 1, 1). Experiment with different values for learning constant so the learning process completes in the fewest number of iterations. During the learning process, print your results to files.

Note:
In a given initial weight set, the last "1" is bias weight.

Report:
1. Save your **Python** program as H211_hard.py and save files with results as H211_hard.txt.
2. Explain your results.

2.1.2 For the previous problem (problem 2.1.1), change the activation function to soft activation function and repeat your training procedure. Start with a weight set (1,1,1,1). Introduce computation of the total error in your program defined as the sum of squares of errors for each pattern. Experiment with learning coefficients to find results in the fewest number of iterations for the TE<0.01.

Report:
1. Save your Python program as name it H212_soft.py and save files with results H212soft.txt.
2. Explain & discuss your results.

2.1.3 For the problem 2.1.1, use soft activation function and modify the script to implement a DELTA training rule. Train the network using initial weights (1,1,1,1). Experiment with the learning constant to produce results in least number of iterations for the TE<0.01.

Report:
1. Save your Python program as H213_delta.py and save files with results as H213_delta.txt.
2. Explain & discuss your results.

Note: please make sure that you read the problem (data) carefully. For example, if the output values are 0 or 1, you should not be using a bipolar activation function.

2.2 Design network that solves XOR (4 pts)

Design a neural network with 2 inputs, 1 output, and 3 neurons, which performs the XOR function:
Note: design means "by hand", not by running an algorithm.

A	B	out
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Report:
1. Provide the network diagram with weights.
2. Explain & discuss your results.

2.3 Multi layer perceptron network

2.3.0 Download Python program (0 pts).

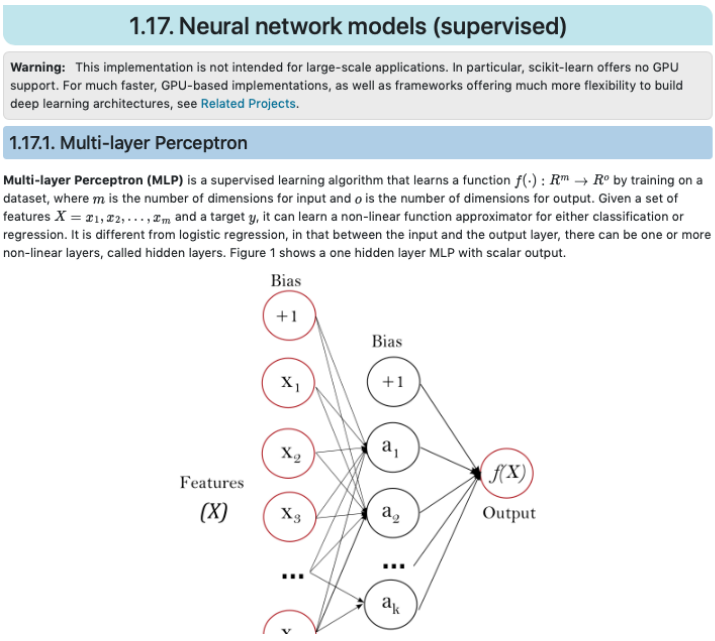
Download Python program *H230_mlp.py* from Canvas. Run it and inspect the output. Make sure you understand the functionalities included in the script. This will be a starting point for the following sections of this homework.
Report: None.

To run Python script, in command prompt type: *python3 script_name.py (*for example, *python3 H230_mlp.py,* to use python 3.0 or above)

```
:~/Desktop$ python3 H230_mlp.py
```

READING MATERIALS:

- Scikit-learn is a Python library that provides implementations of various machine learning algorithms.
- It provides inbuilt functionalities to run neural networks such as Multi-layer Perceptron (MLP)
- Read 1.17.1 to get more details on the MLP algorithm, architecture, advantages, and disadvantages.
- Read 1.17.2 to understand how to use scikit-learn MLP functionality for classification, provide inputs to the algorithm, and predict outcomes.



1.17. Neural network models (supervised)

Warning: This implementation is not intended for large-scale applications. In particular, scikit-learn offers no GPU support. For much faster, GPU-based implementations, as well as frameworks offering much more flexibility to build deep learning architectures, see Related Projects.

1.17.1. Multi-layer Perceptron

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2, \ldots, x_m$ and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 1 shows a one hidden layer MLP with scalar output.

Reading Resource: multi layer perceptron in scikit-learn link.

2.3.1 XOR Problem (2 pts)

Write a program in Python (based on *H230_mlp.py*) to train a MLP implementing the XOR function. Use a soft activation function. Experiment with different parameter values for learning constant (learning_rate in scikit), number of layers and neurons per layer, so the learning process completes in the fewest number of iterations. Provide a screenshot of the best result (i.e. higher accuracy, fewer number of iterations).

In scikit learn, learning constant is defined using two parameters; learning_rate_init and learning_rate.
Ex: clf = MLPClassifier (learning_rate_init=0.001, learning_rate='constant', hidden_layer_sizes=(5, 2))
learning_rate='constant' ('constant' is a constant learning rate given by 'learning_rate_init'.)

XOR function :

A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

test input : 0 1
output : 1

Report:
1. Save your **Python** program as H231_xor.py.
2. Explain your results.

2.4 (4 pts)

2.4.1 Write a program in Python (based on *H230_mlp.py*) to train a MLP implementing the truth table from HW1.4 (homework 1, problem 4). Use a soft activation function. Experiment with different parameter values for learning constant (learning_rate in scikit), number of layers and neurons per layer, so the learning process completes in the fewest number of iterations. Provide a screenshot of the best result (i.e. higher accuracy, fewer number of iterations).

In scikit learn, learning constant is defined using two parameters; learning_rate_init and learning_rate.
Ex: clf = MLPClassifier (learning_rate_init=0.001, learning_rate='constant', hidden_layer_sizes=(5, 2))
learning_rate='constant' ('constant' is a constant learning rate given by 'learning_rate_init'.)

Report:
1. Save your **Python** program as H241.py.
2. Explain your results.

2.5 Extra Credit (3 pts)

Try to solve problem **2.2** with two neurons only. The design should not be achieved by blind guessing (please provide your chain of thoughts).

CMSC 636 Neural Nets and Deep Learning
Spring 2022, Instructor: Dr. Milos Manic, *http://www.people.vcu.edu/~mmanic*
Homework 2

Report:
1. Provide the network diagram with weights.
2. Explain & discuss your results.

Deliverables/Report:

- Once completed, submit to Canvas.
- Compile your results into a **single pdf file**, named: "HWnn_Family_name.pdf", nn being the homework number.
- Scripts can be zipped and submitted as a separate file.
- This assignment is worth 15 points.