**Due: week 12, beginning of class**

**Instruction:**

* Demonstrate your work to your instructor during the class
* Submit your source codes and answers to Moodle
* Your source codes must contain your StudentId and Name

**Student ID: Student Name:**

**Part A: Single layer perceptron and the logic functions**

In this part of the lab, we will find out whether a single layer perceptron can be trained to simulate the basic logic functions of **and**, **or** and **xor** (exclusive or)

**AND OR**

|  |  |  |
| --- | --- | --- |
| **Input 1** | **Input 2** | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

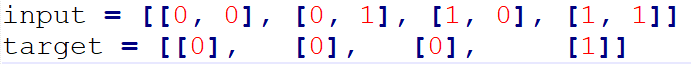
|  |  |  |
| --- | --- | --- |
| **Input 1** | **Input 2** | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**XOR**

|  |  |  |
| --- | --- | --- |
| **Input 1** | **Input 2** | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**The *SLP-logic.py* program**

1. Download the program *SLP-logic.py* from Moodle and study it.
2. Note the input and the output of the sample data are currently defined as:

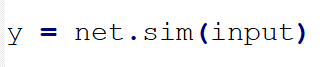


which correspond to the input and output of the **AND** function.

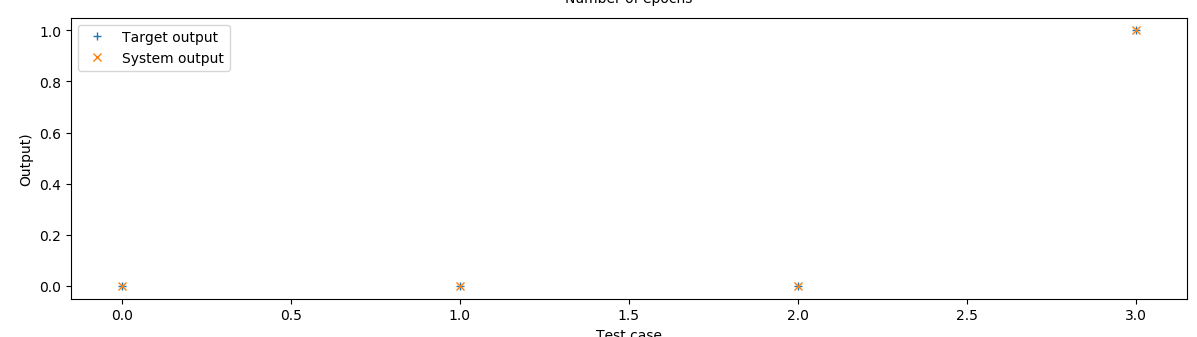
1. The following step in the program defines a single layer perceptron network with two inputs (input data range: 0 to 1), and one output node.



1. The program, then trains the model using the sample data as:
2. The program then tests the model using the command,



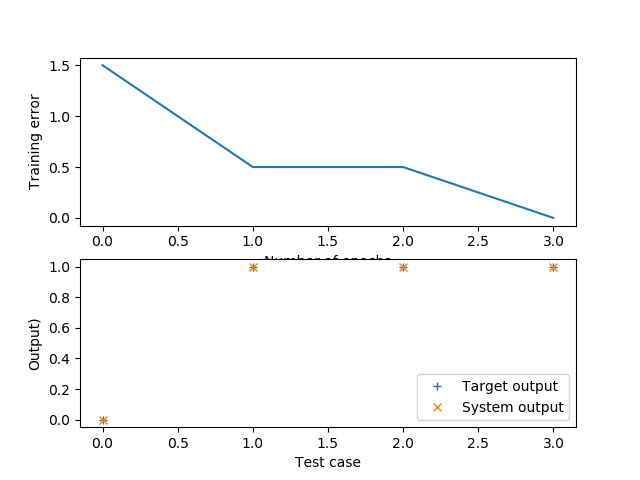
1. The actual outputs and expected target result are shown on a diagram. They should match.



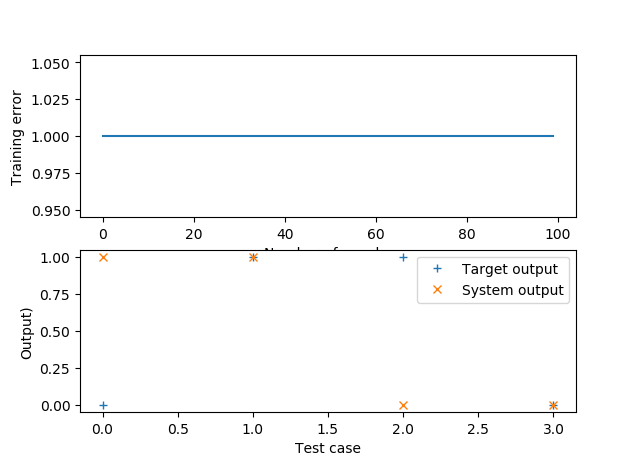
-----------------------------------------------------------------------------------------------------------------**Your task**

Repeat the experiment for the **OR** function and the **XOR** function by modifying the data in step (2) . Which function (**OR** / **XOR**) can be simulated by the single layer perceptron? Copy and paste the resulting figures for comparing system output vs target output below.

**OR**

****

**XOR**

****

**Which function can be simulated by the perceptron?**

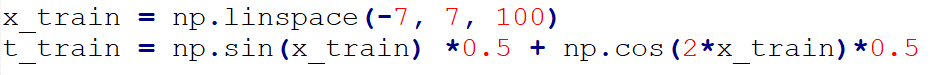
* **AND and OR.**

**Part B: Multi-layer neural network**

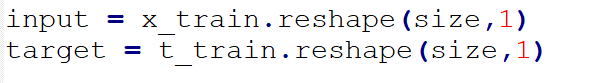
In this part, we will use a multi-layer neural network to simulate the function:

*y* = 0.5 sin(*x*) + 0.5 cos (2*x*)

1. Download the program *multi-layer-NN-Stu.py* from Moodle.
2. The input (*x\_train*) and the expected target outputs (*t\_train*) are defined as:



Note the inputs and the target outputs are converted into column format before they can be used for training for the multi-layer network.



**Your task**

1. You need to create a multi-layer neural network by completing the line of code:
2. Your command should look like

net = nl.net.newff([[*x\_lower*, *x\_upper*]],[*nodes\_in\_layers* ])

where *x\_lower* and *x\_upper* is the upper and lower range of the input values.

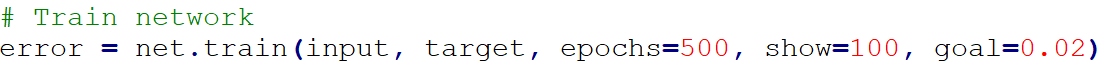
For this exercise, use [-7, 7] as the values (i.e., same as the input data range).

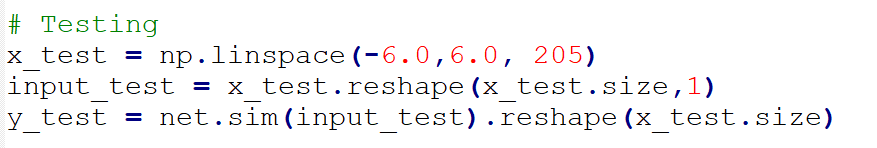
For the *nodes\_in\_layers,* you need to define the number of nodes in each hidden layer and the output layer. For example, you may start with the setting [2, 1], which defines a neural network with one hidden layer and one output layer. There are two nodes in the hidden layer and one node in the output layer.

Your definition should then look like:



1. The program would then train the system using the inputs and target output.



1. After that, then it will test the model using another set of data (x\_test). Note that *x\_test* contains data points that are not included in *x\_train*.
2. A graph of *system output vs target output* for the testing data will be displayed. Do they match well?
3. **Your task:** Modify your of definition the neural network architecture (step 2) by adjusting the number of nodes and number of hidden layer. Find an architecture that can simulate the function well.

State the architecture you used by filling in the function parameter at below:

net = nl.net.newff([-7, 7]], [5, 6, 7, 1])

1. Copy and paste the result figures for comparing system output vs target output below.

