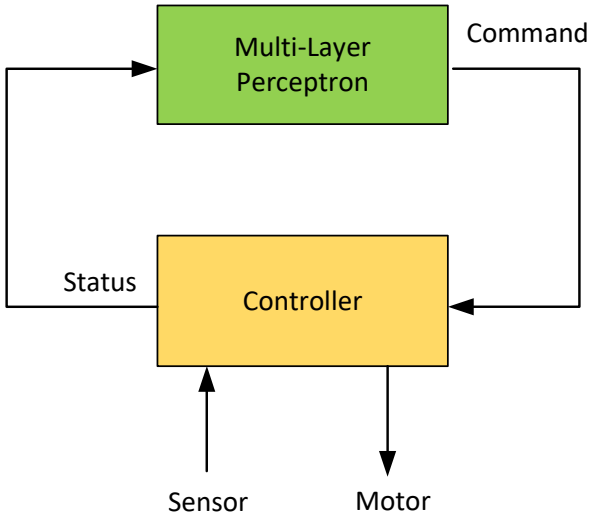
In this project you will build a fully-connected neural network (MLP) to provide some obstacle avoidance capability in Robo-Car . The controller receives distance information from the sensor and sends the status to the MLP. The MLP supplies the command to the controller, which in turn controls the motor.



This pseudo code describes the functionality of the controller.

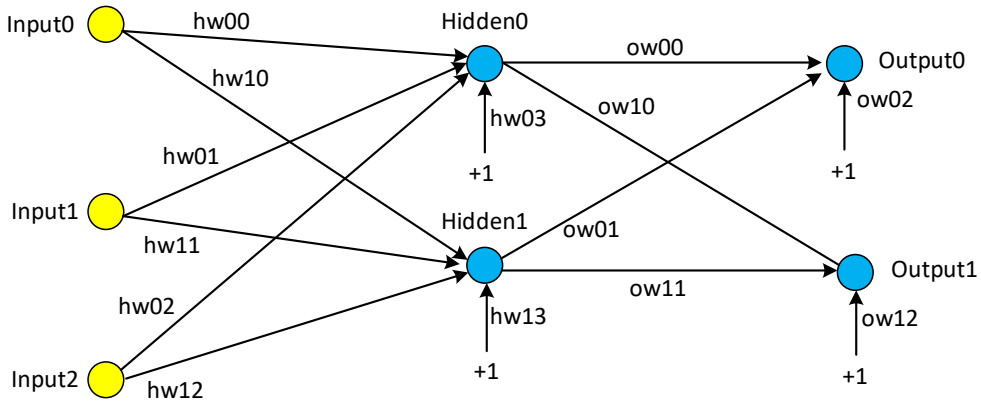
```
select command
       11 (Go Forward):
              if FrontDistance < OBSTACLE_DISTANCE
                     status = 001
              else
                     status = 000
       00 (Stop):
              go backward
              if LeftDistance > RightDistance
                     status = 010
              else
                     status = 100
       10 (Turn Left):
              turn left
              status = 000
       01 (Turn Right):
              turn right
              status = 000
```

In response, the MLP generates the following commands:

Status (Input)			Command (Target)		Remarks
Input 0	Input 1	Input 2	Output 0	Output 1	
0	0	0	1	1	No obstacle detected, go forward
0	0	1	0	0	Obstacle, stop, check left & right distances
0	1	0	1	0	Left distance > right distance → turn left
1	0	0	0	1	Left distance < right distance → turn right

Table 1

The MLP has 3 input nodes, 1 hidden layer with 2 neurons and 2 output neurons. All neurons use the sigmoid function as their activation functions.



hw = hidden weight

ow = output weight

With back-propagation use Table 1 to train the MLP.

Run for 15,000 epochs or until the error is less than 0.005. The error is defined as

$$\varepsilon = \frac{1}{2}\sum_{k=1}^{2} e_k^2 = \frac{1}{2}\sum_{k=1}^{2} (d_k - y_k)^2$$

Where for an output neuron k, d_k is the target value (command output) and y_k is the predicted value.

Randomize the dataset with each epoch.

Write the hidden weights and the output weights to a *csv* file called *MLP_weights.csv*. Your code should automatically download this file to a local directory. You must organize the hidden weights and the output weights in the *csv* file as follows:

hw00	hw01	hw02	hw03
hw10	hw11	hw12	hw13
ow00	ow01	ow02	
ow10	ow11	ow12	

To read the file and verify the output of the MLP, the TA will run *Verify_Weights.ipynb*, which will produce the following output:

```
Training Pattern:  0
 Input [0 0 0]  Target [1 1]  Output [[0.99389198 0.97867152]]

 Training Pattern:  1
 Input [0 0 1]  Target [0 0]  Output [[0.00456837 0.01349511]]

 Training Pattern:  2
 Input [0 1 0]  Target [1 0]  Output [[0.98166315 0.03655929]]

 Training Pattern:  3
 Input [1 0 0]  Target [0 1]  Output [[0.04189467 0.97473363]]
```

Note this is just to illustrate what you can expect. You should run it but don't try to match the values of the output neurons.

Table 2 shows the expected value of an output neuron for RoboCar to work properly.

Target	Output neuron
1	> 0.80
0	< 0.20

Table 2

- Submit your *iPython* code.
- *Discussing this project with other students is highly recommended but you have to submit your own solution.*