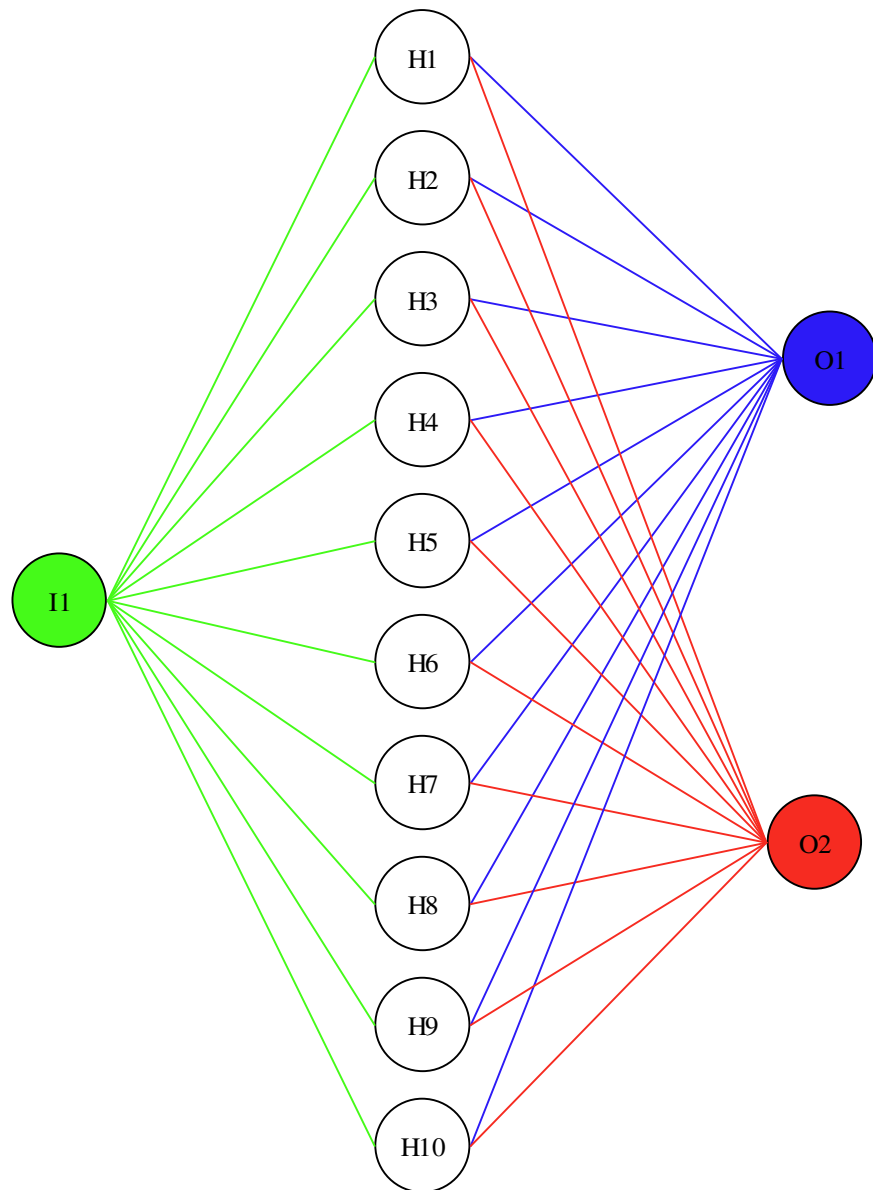


How the Hovercraft Control Logic Works

The Hovercraft Control Logic System's (HCLS) brain is the Obstacle Avoidance Logic (OAL). The OAL used here is an Artificial Neural Network (ANN). The ANN is trained how to behave, instead of programmed like conventional logic. Lets take a look at exactly how the hovercraft's OAL works.

How the Neural Network Recalls Learnt Information

The key to the ANN is the connections between each neuron. The ANN used on this hovercraft is called a feed-forward neural network, as data only goes from input to output, and does not feedback. This diagram shows the actual ANN running on the hovercraft, and has one input neuron, 10 hidden neurons and two output neurons.

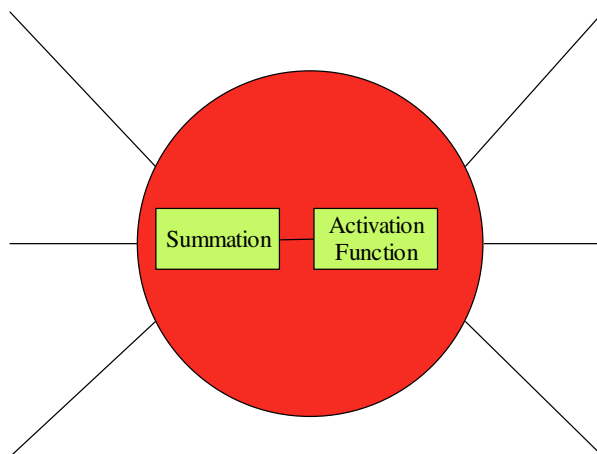


Each of those connections between one neuron and another has a certain weighted value. This chart shows the weighted value for each connection.

Neuron	Neuron	Weight
I1	H1	-0.704894
I1	H2	-6.526604
I1	H3	0.568346
I1	H4	4.448008
I1	H5	2.076094
I1	H6	2.238839
I1	H7	2.374001
I1	H8	7.698660
I1	H9	2.429901
I1	H10	2.426541
H1	O1	2.511209
H1	O2	-2.440218
H2	O1	13.311865
H2	O2	13.178253
H3	O1	3.122861

Neuron	Neuron	Weight
H3	O2	-1.016410
H4	O1	1.125672
H4	O2	-2.862084
H5	O1	-3.051573
H5	O2	1.669890
H6	O1	-3.647110
H6	O2	3.076672
H7	O1	-2.520317
H7	O2	4.681576
H8	O1	3.402647
H8	O2	-11.532937
H9	O1	-2.229351
H9	O2	5.593643
H10	O1	-1.643753
H10	O2	5.045427

The inside of an artificial neuron is shown in this diagram.



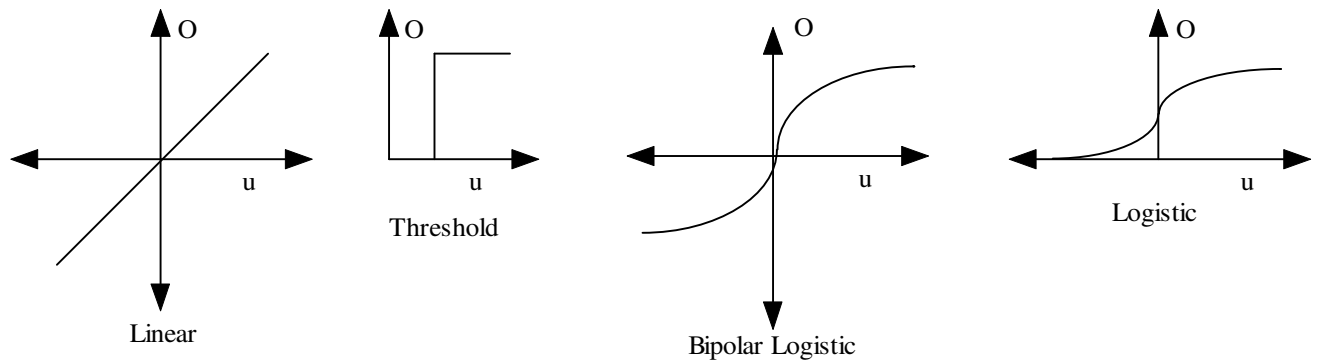
The inputs to the neuron are first passed through a summation function. The summation function is simply:

$$u = (o_1 \bullet w_1) + (o_2 \bullet w_2) + (o_n \bullet w_n)$$

Where O is the output from the connected neuron that is presented as the input to this neuron, and w is the weight of that connection. Once all the weights have been calculated, the output of that is passed into the activation function. The activation function is the key to the neural network, and there are several different types.

Here are a few of the main types of activation functions:

Note: Approximated Graphs



The input and output for all these functions are between -1 to 1 . Several of them are limited from 0 to 1 though.

Linear: The linear activation function is one of the simplest. It is very fast, however it suffers in that many problems it is unable to solve.

Threshold: Threshold is a simple activation function. When the input is below a certain value, the output is minimum. When the input is above a certain value, the output goes to maximum.

Bipolar Logistic: Powerful and easy to train, the logistic functions are among the most popular in use. Bipolar logistic allows the input and output go between -1 to 1 .

Logistic: Again, powerful and easy to train, however logistic (as opposed to bipolar logistic) only allows an input between 0 to 1 .

This hovercraft uses a logistic function on its artificial neural network. This function is represented by the function $output = 1 / (1 + \exp^{-u})$.

Here is a quick example, using the actual hovercraft's weight table.

1. It has the value 0.743 presented to it. In the neural network is a simple input neuron, and doesn't actually have any summation or activation function associated with it.
2. The value 0.743 is sent to the input of all the hidden neurons.
3. The summation function is applied to this input for each neuron. The table of weights is recalled, and the appropriate weight value is read from the table. Then this weight value is multiplied by the input value, which is 0.743 .

For example the connection between I1 and H2 has the weight value -6.526604, and 0.743 multiplied by -6.526604 is equal to -4.849266772.

4. The output of the summation function is applied to the activation function, which in this neural network is $output = 1/(1 + \exp^{-u})$, where u is the output of the summation function.

Using the previously calculated value of -6.526604, the output of hidden neuron H2 would be 0.007773223.

5. The output of every hidden layer neuron is applied to the input of both output layer neurons. The summation function is used first, its calculate the product of the output of each hidden layer neuron multiplied by the appropriate weight, and all the products of the weight multiplied by the input are added together. This one number is fed into the activation function, and the output of the activation function is the output of the neural network.

For example if the output neuron 1 had 0.847 inputted to it by H1 and 0.007773223 inputted to it by H2, then the output of the summation function would be

$(0.847 \bullet 2.511209) + (0.007773223 \bullet 13.311865) = 2.230470118$ Then this value is put into the activation function $output = 1/(1 + \exp^{-u})$. Then the output of the activation function would be 0.902952562. This is assuming the output neuron only had two inputs to it; in reality it actually has ten.

The true power of a neural network is in training it. The neural network described above only has the ability to recall a previously learnt weight file, but not too actually retrain itself.

How Neural Networks Learn

Teaching a neural network can be a fairly complex process. To begin with, a lesson file is created. The lesson file has the different lessons to be taught to the neural network. A lesson file normally takes the form of:

<inputs> <outputs>

So for example the lesson file used to train the hovercraft could look like:

```
7 1 2
.1 1 1
.2 .8 .9
.3 .4 .7
.4 .2 .6
.5 .1 .6
.7 0.05 .8
1 0 .9
```

The first line is added for the training program, means there are 7 lessons, 1 input, and 2 outputs. The remaining lines follow the format of

<distance> <speed> <turn>

Where all the numbers are values scaled down to be between 0 to 1. In the hovercraft the distance is from 0 to 500cm (0 = 0cm, 0.5 = 250, 1 = 500cm). Speed is approximately equal to 0 to 100% of maximum power (0 = 0% speed, 0.5 = 50% speed, 1 = 100% speed). Turn is an absolute value, and does not include a direction. This value only references how much to turn (0 = no turn, 0.5 = some turn, 1 = full turn), it is up to the Turn Direction Logic to decide which direction to turn.

This lesson file is what is taught to the neural network. The technique used in this neural network for training is called “back propagation”. Back propagation’s basic principle is to first pass the input data (training) through the neural network. Then the output of the neural network is compared with what the training file says the output should be. This error is passed backwards through the neural network. This error is used to calculate the weight changes that should occur. This process continues until the error is low enough to not be a concern.

However convergence can be difficult on a neural network. This means that it won’t always be possible to make the error very low for every input. In situations where it doesn’t seem possible, sometimes increasing the number of neurons helps. Also there is a danger of “over-training” the neural network. Once the error reaches its lowest value, it is possible that the error will actually increase! This is why the actual training is a very complex process, as there are many variables to control and consider.