**SOFT COMPUTING ASSIGNMENT**

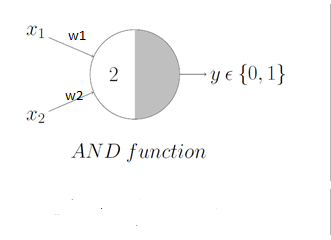
---------------------------------------------------------------------------------------------------------

Jeevan Koshy ~ 1740256

---------------------------------------------------------------------------------------------------------

**AND Gate using MP neuron**

One such application of the MP neuron is that it can be used to represent Boolean functions. Let x1, x2 be 2 inputs that goes in the M-P neuron. The neuron maybe divided into 2 parts. The first part takes an input and then performs operations on the input. Based on the aggregated value, the second part makes a decision.



**X1 X2 Y**

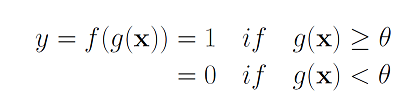
|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

From the above table, it is clear that if x1 & x2 are 1 only then will the neuron fire or else it won’t.

The net input can be calculated using the formula and the output will be given depending on the threshold value when the net input is passed through an activation function -

Input function: ∑ wixi   
Output: func (∑ wixi)

Eg. If we consider the activation function given by



For weights (say w1 = 0.2, w2 = 0.4) & Θ = 0.5;

If x1, x2 = 1, 1; Net input = ∑ wixi = 0.2 \* 1 + 0.4 \* 1   
 = 0.2 + 0.4   
 = 0.6

Therefore,   
The activation function yields the result ~ 1; when both inputs are 1  
 0; otherwise

---------------------------------------------------------------------------------------------------------

**Perceptron Algorithm**

*STEP 1:* Start

*STEP 2:* Initialise weights & biases

*STEP 3:* Set learning rate α (0 or 1).

*STEP 4:* Activate inputs

*STEP 5:* Calculate total net inputs

*STEP 6:* Apply activation function

*STEP 7:* Check whether output is equal to threshold value.

*STEP 8:* Else assign new output as previous old value.

*STEP 9:* If weight changes, repeat steps 4 to 8.

*STEP 10:* Stop

---------------------------------------------------------------------------------------------------------

**Error Back Propagation**

Computing the error to adjust the weight.

*STEP 1:* Start

*STEP 2:* Set weights to small random values

*STEP 3:* Peform algorithm for each training pair

*STEP 4:* Recieve inputs and transmit to hidden unit

*STEP 5:* Calculate output in the hidden layer

*STEP 6:* Apply activation function

*STEP 7:* Calculate signal output from the output layer

*STEP 8:* Else assign new output as previous old value.

*STEP 9:* If weight changes, repeat steps 4 to 8.

*STEP 10:* Stop

---------------------------------------------------------------------------------------------------------