McCulloch-Pitts Neural Model

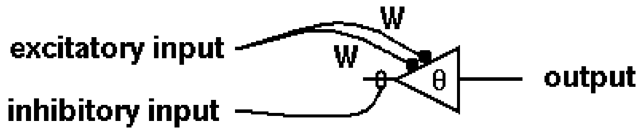
In 1943 two electrical engineers, Warren MMcCulloch and Walter Pitts, published the first paper describing what we would call a neural network. Their "neurons" operated under the following assumptions:

1. They are binary devices ($V_i = [0,1]$)
2. Each neuron has a fixed threshold, theta
3. The neuron receives inputs from excitatory synapses, all having identical weights. (However it my receive multiple inputs from the same source, so the excitatory weights are effectively positive integers.)
4. Inhibitory inputs have an absolute veto power over any excitatory inputs.
5. At each time step the neurons are simultaneously (synchronously) updated by summing the weighted excitatory inputs and setting the output (V_i) to 1 if the sum is greater than or equal to the threshold AND if the neuron receives no inhibitory input.

We can summarize these rules with the McCullough-Pitts output rule

$$V_i = \begin{cases} 1 & : \quad \sum_j W V_j \geq \theta \text{ AND no inhibition} \\ 0 & : \quad \text{otherwise} \end{cases}$$

and the diagram

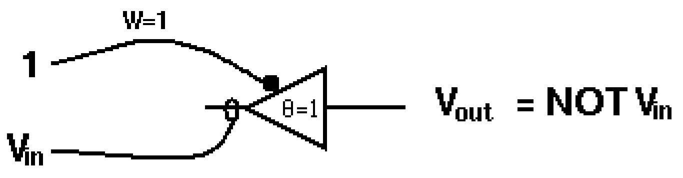


Using this scheme we can figure out how to implement any Boolean logic function. As you probably know, with a NOT function and either an OR or an AND, you can build up XOR's, adders, shift registers, and anything you need to perform computation.

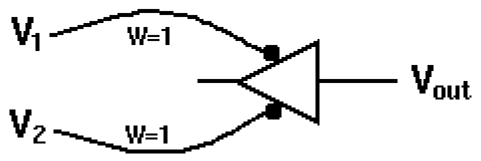
We represent the output for various inputs as a **truth table**, where 0 = FALSE, and 1 = TRUE. You should verify that when W = 1 and theta = 1, we get the truth table for the logical NOT,

```
Vin  |  Vout
-----+------
  1  |   0
  0  |   1
```

by using this circuit:



With two excitatory inputs V_1 and V_2, and W =1, we can get either an OR or an AND, depending on the value of theta:



if $\theta = 1 \Longrightarrow V_{out} = V_1 \text{ OR } V_2$

if $\theta = 2 \Longrightarrow V_{out} = V_1 \text{ AND } V_2$

Can you verify that with these weights and thresholds, the various possible inputs for V_1 and V_2 result in this table?

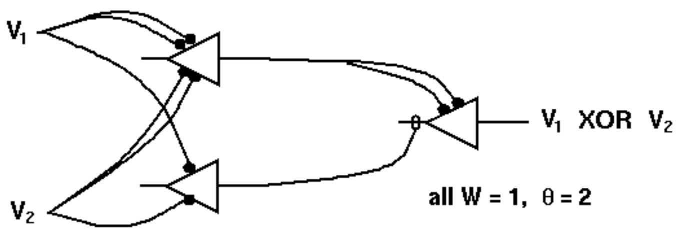
V1	V2	OR	AND
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

The **exclusive OR** (XOR) has the truth table:

V1	V2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

(Note that this is also a "1 bit adder".)

It cannot be represented with a single neuron, but the relationship
XOR = (V_1 OR V_2) AND NOT (V_1 AND V_2) suggests that it can be represented with the network



Exercise: Explain to your own satisfaction that this generates the correct output for the four combinations of inputs. What computation is being made by each of the three "neurons"?

These results were very encouraging, but these networks displayed no learning. They were essentially "hard-wired" logic devices. One had to figure out the weights and connect up the neurons in the appropriate manner to perform the desired computation. Thus there is no real advantage over any conventional digital logic circuit. Their main importance was that they showed that networks of simple neuron-like elements could compute