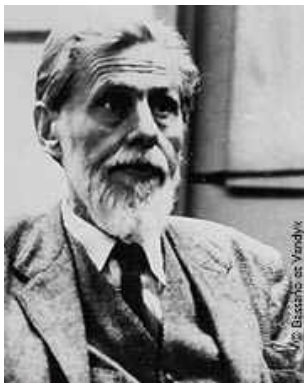
## The McCulloch-Pitts Neuron

Nowadays the McCulloch-Pitts neuron tends to be overlooked in favour of simpler neuronal models but they were and are still important. They proved that something that behaved like a biological neuron was capable of computation and early computer designers often thought in terms of them.

Before the neural network algorithms in use today were devised, there was an alternative. It was invented in 1943 by neurophysiologist Warren McCulloch and logician Walter Pitts. Now networks of the McCulloch-Pitts type tend to be overlooked in favour of “gradient descent” type neural networks and this is a shame. McCulloch-Pitts neurons are more like the sort of approach we see today in neuromorphic chips where neurons are used as computational units.



*Warren McCulloch ; Walter Pitts*

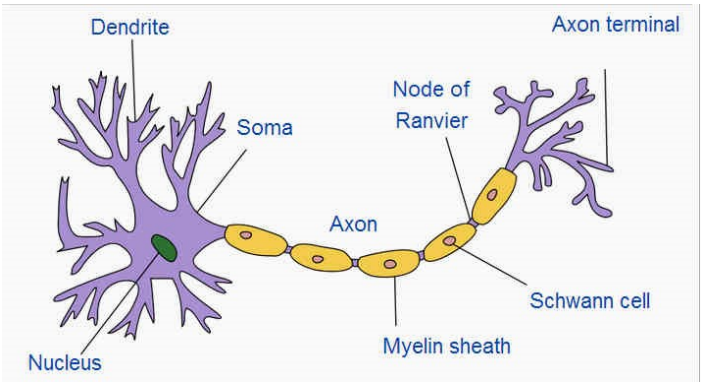
What is interesting about the McCulloch-Pitts model of a neural network is that it can be used as the components of computer-like systems.

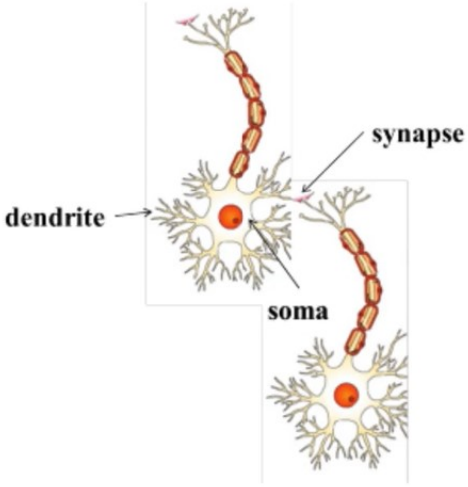
That is, where neural networks are commonly used to learn something, a McCulloch-Pitts neuron is constructed to do a particular job.

OK, I have to admit that in many cases the same job can be done just as well using traditional Boolean components, but it is interesting to see how it all works using components which are closer to “biological” components. It gives you an idea, although probably not the whole idea, of how the brain might just be a computer.

## The Biological Neuron

The basic idea of a McCulloch-Pitts model is to use components which have some of the characteristics of real neurons.




A real neuron has a number of inputs, the dendrites, which are “excitatory” and some which are “inhibitory”. What the neuron does depends on the sum of inputs. The excitatory inputs tend to make the cell fire and the inhibitory inputs make is not fire – i.e. pass on the signal.

When the cell fires a signal propagates down the axon which acts like the output wire and then is applied to the inputs of other neurons via the axon terminals.

You can think of it as a sort of battle to control the neuron with the excitatory inputs pushing it to fire and the inhibitory inputs stopping it.

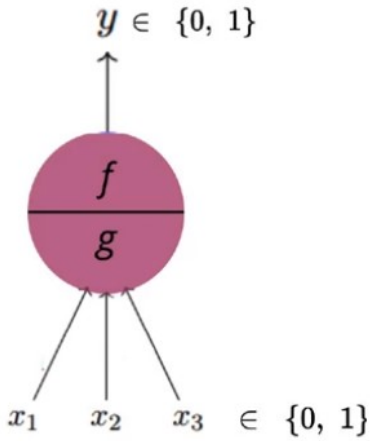
There are lots of complications to this basic model – in particular a neuron fires a train of pulses rather than just changing its state - but the idea of inhibitory fighting excitatory seems to be the key behaviour.

How can we create an easier to use model of the neuron?

## McCulloch-Pitts Neuron Model

MP Neuron Model introduced by Warren McCulloch and Walter Pitts in 1943. MP neuron model is also known as linear threshold gate model.

## Mathematical Model




Fig—2 Simple representation of MP Neuron Model
The function (soma) is actually split into two parts: g—The aggregates the inputs to a single numeric value and the function f produces the output of this neuron by taking the output of the g as the input i,e.. a single value as its argument. The function f will output the value 1 if the aggregation performed by the function g is greater than some threshold else it will return 0.

The inputs x1, x2, ….. xn for the MP Neuron can only take boolean values and the inputs can be inhibitory or excitatory. Inhibitory inputs can have maximum effect on the decision-making process of the model. In some cases, inhibitory inputs can influence the final outcome of the model.

$y = 0$ if any $x_i$ is inhibitory, else

$$g(x_1, x_2, ...x_n) = g(x) = \sum_{i=1}^{n} x_i$$

$$y = f(g(x)) = 1 \text{ if } g(x) \geq b$$

$$= 0 \text{ if } g(x) < b$$

Fig—3 Mathematical representation
For example, I can predict my own decision of whether I would like to watch a movie in a nearby IMAX theater tomorrow or not using an MP Neuron Model. All the inputs to the model are boolean i.e., [0,1] and from the Fig—2 we can see that output from the model will also be boolean. (0—Not going to movie, 1—going to the movie)

Inputs for the above problem could be

x1—IsRainingTomorrow (Whether it's going to rain tomorrow or not)
x2—IsScifiMovie (I like science fiction movies)

x3—IsSickTomorrow (Whether I am going to be sick tomorrow or not depends on any symptoms, eg: fever)
x4—IsDirectedByNolan (Movie directed by Christopher Nolan or not.)
etc.…
In this scenario, if x3—IsSickTomorrow is equal to 1, then the output will always be 0. If I am not feeling well on the day of the movie then no matter whoever is the actor or director of the movie, I wouldn't be going for a movie.

**a Simgle Model**

What we need is a model of the neuron that doesn't capture all of its properties and behaviour but just enough to capture the way it performs compuation. So what we have to do is throw away aspects of the way the cell behaves in an effort to simplify without destroying its action.

First let's just have a cell that gives out a binary state – zero or one, on or off.

The inputs then carry a binary signal and the only thing that matters is the number of "on" signals on the excitatory versus the inhibitory inputs.

We have essentially a "threshold" gate model of the neuron.

This turns out to be a bit too complicated to analyse so we make one more big simplification – the inhibitory input overrides the excitatory.

That is, if an inhibitory input is on the cell cannot fire (i.e. it is off) no matter what the excitatory inputs are doing. With this small change things are made a lot easier and we can now define precisely what a McCulloch-Pitts neuron is:

1. a cell which can output a 0 or a 1
2. a number of excitatory inputs
3. a single inhibitory input
4. a threshold value

The rule for its operation is that at time *t* it looks at its excitatory inputs and counts up the number of ones present. If the count is equal to or greater than the threshold AND the inhibitory input is zero then at time *t+1* the cell outputs a one, otherwise it outputs a zero.

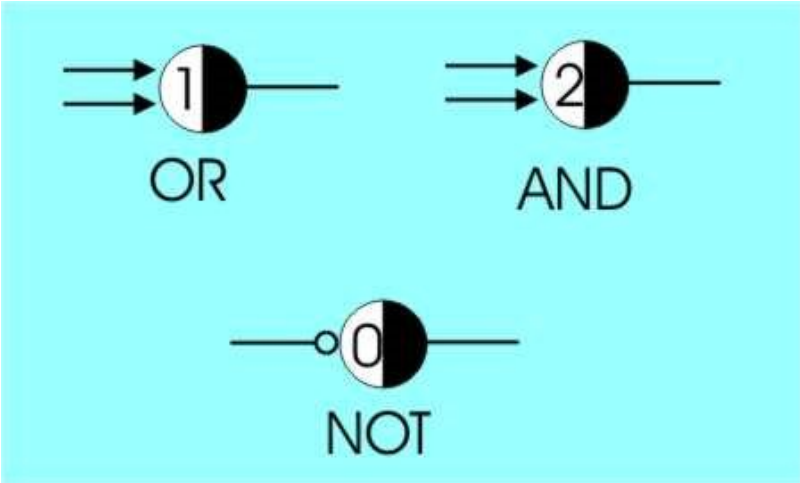
Notice that this rule is more complicated than you might think at first and it does give rise to behaviours that are quite complicated.

The key factor that you might just overlook is that we are using sequential logic that involves time. A clock is used to synchronise all of the cells in a network and they change state at each clock pulse. This is necessary to stop infinite period oscillation in some types of circuit.

**Standard Logic**

I'm going to use the symbols and notation introduced by Marvin Minsky. Minsky uses a symbol that shows a cell as a circle with its threshold written in.

It is very easy to invent equivalents of the standard Boolean gates. For example, it is very easy to work out how to make OR, AND and NOT.




*The basic gates of Boolean logic as McCulloch-Pitts Neurons*

You should be able to see that the truth tables for each of these cells is equivalent to the corresponding Boolean gate.

For example, for the AND cell, one input at one doesn't fire the cell but two and only two at one does fire the cell – i.e. an AND gate as promised. The OR gate fires if either or both of its inputs are one because its threshold is one.

The only tricky one is the NOT gate and you can see here that a single inhibitory input (shown with a circle at the end) combined with a threshold of zero gives the desired result. When the inhibitor is off the threshold is equalled, i.e. zero excitatory inputs and the cell give out a one. If the inhibitor is on then the cell is obviously off – a NOT gate as promised.

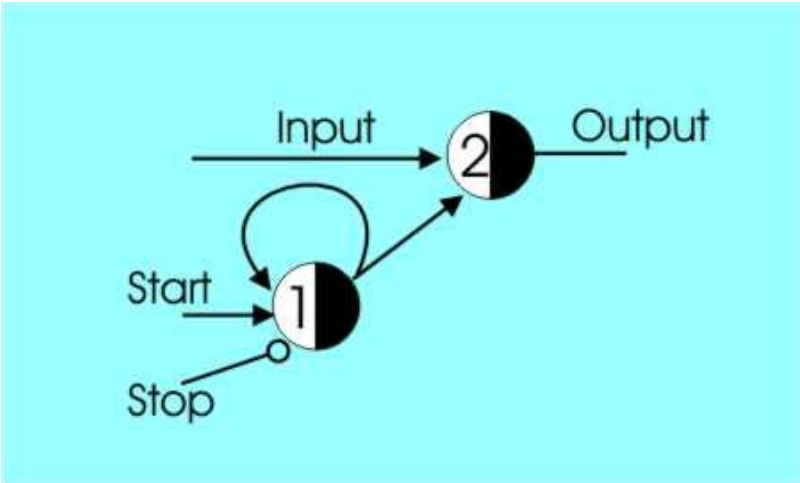
**More Than Boolean?**

Given we can make up the basic gates of Boolean logic it is clear that that McCulloch-Pitts networks are up to building anything that can be built with standard logic gates, but are they capable of anything more?

The answer is yes and the reason is partly that they have time built into their specification and partly because they are threshold gates.

For example, a delay gate can be built using a single excitatory input with a threshold of one. Stacking such gates up provides a delay of more than one time unit. Delaying a signal is the same as remembering it for *n* time periods, so cells have memory!

If you would like a clearer example of the memory inherent in McCulloch-Pitts cells have a look at the following simple latch.




*A latch*

The top cell passes the input to the output if it has a one from the other cell. You can see that the feedback loop in the lower cell keeps it switched on once it is on irrespective of the input on the Start line. Once on the only way to switch it off is to put a one on the Stop line which, being an inhibitor, turns everything off.

explor the MP Neuron with URL:
http://www.mind.ilstu.edu/curriculum/modOverview.php?modGUI=212