Introduction to Artificial Neural Networks

The Machine Learning Tsunami:- In 2006, Geoffrey Hinton et al. published a paper showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this technique called “Deep Learning.”

ANNs are at the very core of Deep Learning. They are versatile, powerful, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks such as classifying billions of images (e.g., Google Images , helps Google for data mining ), powering speech recognition services (e.g., Apple’s Siri), recommending the best videos to watch to hundreds of millions of users every day (e.g., YouTube , Netflix), in 1997 IBM’s Deep Blue beats the world chess champion Gary Kasprov , in 2011 IBM’s Watson(Watson is a question-answering computer system capable of answering questions posed in natural language , Watson was named after IBM's founder and first CEO, industrialist Thomas J. Watson.) outplayed two humans at the popular television quiz show **Jeopardy**! The computer's opponents weren't merely two humans—they were the two all-time best **Jeopardy**! champions, ever, learning to beat the world champion lee sedol(18 times world go champion) at the game of Go(a 3000 years old Chinese game which is more and more complex than chess ) (DeepMind’s AlphaGo , 2015 , The Go Game got 10 to the power of 170 **possible** board configurations ,  this is makes Alpha Go to have a combination more than **the number of atoms** in the known universe(10^78 – 10^82). This makes the game of **Go** more complex than chess.) , flagging fake news , self driving cars and predicting Earth Quick and more and more.

An ANN is a Machine Learning model inspired by the networks of biological neurons found in our brains. However, although planes were inspired by birds, they don’t have to flap their wings. Similarly, ANNs have gradually become quite different from their biological cousins. Some researchers even argue that we should drop the biological analogy altogether (e.g., by saying “units” rather than “neurons”), lest we restrict our creativity to biologically plausible systems.

Surprisingly, ANNs have been around for quite a while: they were first introduced back in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts. In their landmark paper2 “A Logical Calculus of Ideas Immanent in Nervous Activity,” McCulloch and Pitts presented a simplified computational model of how biological neurons might work together in animal brains to perform complex computations using propositional logic. This was the first artificial neural network architecture. Since then many other architectures have been invented, as we will see.

The early successes of ANNs led to the widespread belief that we would soon be con‐ versing with truly intelligent machines. When it became clear in the 1960s(the 1 layer perceptron) that this promise would go unfulfilled (at least for quite a while), funding flew elsewhere, and ANNs entered a long winter. In the early 1980s, new architectures were invented and better training techniques were developed, sparking a revival of interest in connectionism (the study of neural networks). But progress was slow, and by the 1990s other powerful Machine Learning techniques were invented, such as Support Vector Machines . These techniques seemed to offer better results and stronger theoretical foundations than ANNs, so once again the study of neural networks was put on hold.

Training a deep neural net was widely considered impossible at the time, and most researchers had abandoned the idea in the late 1990’s(Support Machine Learning Algorithms are the most widely used until 2006 ). This paper(Geoffrey Hinton Hand written digits recognition ) revived the interest of the scientific community, and before long many new papers demonstrated that Deep Learning was not only possible, but capable of mind blowing achievements that no other Machine Learning (ML) technique could hope to match (with the help of tremendous computing power and great amounts of data). This enthusiasm soon extended to many other areas of Machine Learning.

A decade or so later, Machine Learning has conquered the industry: it is at the heart of much of the magic in today’s high-tech products, ranking your web search results, powering your smartphone’s speech recognition, recommending videos, and beating the world champion at the game of Go. Before you know it, it will be driving your car.

We are now witnessing yet another wave of interest in ANNs. Will this wave die out like the previous ones did? Well, here are a few good reasons to believe that this time is different and that the renewed interest in ANNs will have a much more profound impact on our lives:

There is now a huge quantity of data available to train neural networks(which is one of the largest problem in the late 1990’s), and ANNs frequently outperform other ML techniques on very large and complex problems.

The tremendous increase in computing power since the 1990s now makes it possible to train large neural networks in a reasonable amount of time. This is in part due to Moore’s law (the number of components in integrated circuits has doubled about every 2 years over the last 50 years), but also thanks to the gaming industry, which has stimulated the production of powerful GPU cards by the mil‐ lions. Moreover, cloud platforms have made this power accessible to everyone.

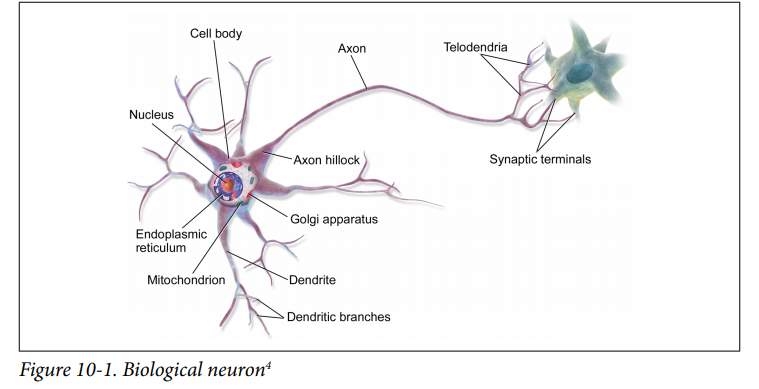
The training algorithms have been improved. To be fair they are only slightly different from the ones used in the 1990’s, but these relatively small tweaks have had a huge positive impact.

ANNs seem to have entered a virtuous circle of funding and progress. Amazing products based on ANNs regularly make the headline news, which pulls more and more attention and funding toward them, resulting in more and more progress and even more amazing products.

Modern deep learning provides a very powerful framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training examples. Other tasks, that can not be described as associating one vector to another, or that are difficult enough that a person would require time to think and reflect in order to accomplish the task, remain beyond the scope of deep learning for now.

Biological Neurons

Before we discuss about artificial neurons, let’s take a quick look at a biological neuron. Neurons are un-usual looking cells mostly found in animals brain which has a number of input wires called Dendrite(accept input from other locations) and also has an output wire called Axons(to send signals or messages to other neurons) with a cell body containing the nucleus. Neurons are generally a computational units that get a number of input wires with their dendrites , so some computations with their Nucleus and send outputs with various Axons to other nucleus. Near its extremity the axon splits off into many branches called telodendria, and at the tip of these branches are minuscule structures called synaptic terminals (or simply synapses), which are connected to the dendrites or cell bodies of other neurons. One output of the Axons will connect to the input wire Dendrites of other Neurons. There are about 100billion(10^11) Neurons on our brain. Each neuron is typically connected to thousands of other neurons, so that it is estimated that there are about 100 trillion (= 10^14) synapses within the brain. Biological neurons produce short electrical impulses called action potentials (APs, or just signals) in order to communicate with each other and travel along the axons and make the synapses release chemical signals called neurotransmitters. When a neuron receives a sufficient amount of these neurotransmitters within a few milliseconds, it fires its own electrical impulses (actually, it depends on the neurotransmitters, as some of them inhibit the neuron from firing). Each neuron can be viewed as a separate processor, performing a very simple computation: This makes the brain a massively parallel computer made up of 10^11 processing elements. If that is all there is to the brain, then we should be able to model it inside a computer and end up with animal or human intelligence inside a computer. This is the view of strong AI.



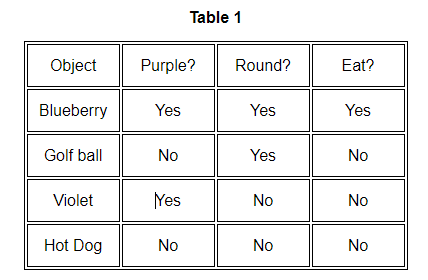
Thus, individual biological neurons seem to behave in a rather simple way, but they are organized in a vast network of billions, with each neuron typically connected to thousands of other neurons. Highly complex computations can be performed by a network of fairly simple neurons. In the context of Machine Learning, the phrase “neural networks” generally refers to ANNs, not BNNs.

McCulloch and Pitts Neurons

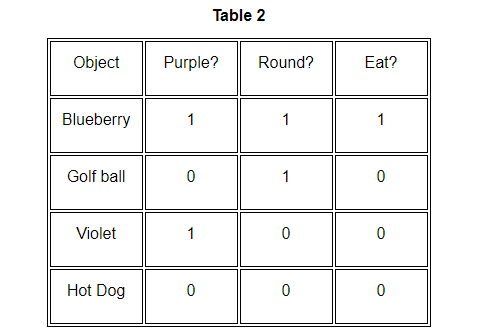
In 1943 Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, published "A logical calculus of the ideas immanent in nervous activity" in the In this paper McCulloch and Pitts tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together. These basic brain cells are called neurons, and McCulloch and Pitts gave a highly simplified model of a neuron in their paper. The McCulloch and Pitts model of a neuron, which we will call an MCP neuron for short, has made an important contribution to the development of artificial neural networks -- which model key features of biological neurons. The MCP neuron is not a real neuron; it's only a highly simplified model. We must be very careful in drawing conclusions about real neurons based on properties of MCP neurons.

Some MCP neuron examples

In order to understand MCP neurons, let's look at an example. Suppose there is a neuron in a bird's brain that has two receivers, which are connected somehow to the bird's eyes. If the bird sees a round object, a signal is sent to the first receiver. But if any other shape is seen, no signal is sent. So the first receiver is a roundness detector. If the bird sees a purple object, a signal is sent to the second receiver of the neuron. But if the object is any other color, then no signal is sent. So the second receiver is a purple detector. Notice that for either receiver there is a question that can be answered "yes" or "no," and a signal is only sent if the answer is "yes." The first receiver corresponds to the question "Is the object round?" The second receiver corresponds to the question "Is the object purple?" We would like to produce an MCP neuron that will tell the bird to eat a blueberry, but to avoid eating red berries or purple violets. In other words, we want the MCP neuron to send an "eat" signal if the object is both round and purple, but the MCP neuron will send no signal if the object is either not round or not purple, or neither round nor purple. So the bird will only eat an object if the MCP neuron sends a signal. If no signal is sent, then the bird will not eat the object. Here is a table that summarizes how the MCP neuron would work in several cases.



Notice that all the signals sent to the MCP neuron and the signal that it sends out are all "yes" or "no" signals. This "all or nothing" feature is one of the assumptions that McCulloch and Pitts made about the workings of a real neuron. They also assumed that somehow a real neuron "adds" the signals from all its receivers, and it decides whether to send out a "yes" or "no" signal based on the total of the signals it receives. If the total of the received signals is high enough, the neuron sends out a "yes" signal; otherwise, the neuron sends a "no" signal. In order to "add" the signals that the MCP neuron is receiving, we will use the number 1 for a "yes" and the number 0 for a "no." Then Table 1 will now look like this.



Now we need a way to decide if the total of the received signals is "high enough." The way McCulloch and Pitts did this is to use a number they called a **threshold**. So what is a threshold, and how does it work? Every MCP neuron has its own threshold that it compares with the total of the signals it has received. If the total is bigger than or equal to the threshold, then the MCP neuron will send out a 1 (i.e. a "yes" signal). If the total is less than the threshold, then the MCP neuron will send out a 0 (i.e. a "no" signal). So the MCP neuron is answering the question "Is the sum of the signals I received greater than or equal to my threshold?"

In order to see how this threshold idea works, let's suppose that we have an MCP neuron with two receivers connected to a bird's eyes. The one receiver is a roundness detector and the other is a purple detector, just as we had in the example above. Since we want the neuron to instruct the bird to eat blueberries but not golf ball, violets or hotdogs we need a threshold high enough so that it requires that both of the two properties are present. Let's try a threshold of "2" and see if that doesn't work.

If the bird sees a blueberry, then the purple detector sends a 1 and the roundness detector sends a 1. So our MCP neuron adds these signals to get a **combined input** of 1+1 =2. Now our MCP neuron takes this total input of 2 and compares it to its threshold of 2. Because the total input (= 2) is greater than or equal to the threshold (=2), the MCP neuron will send an output of 1 (which means, "EAT").

We should now be familiar with the basic workings of an MCP neuron. Your next task is to plug in different values for the **threshold** to produce different types of behavior in the bird. Where would you set the threshold so the bird would eat 3 of the 4 objects? none of the objects? all four of the objects? In the interactive exercises below, you will be asked to set the threshold so as to cause the bird to behave in different ways. There are three separate tasks that you can perform. We encourage you to do all of them.

Let's summarize what we know so far. The behavior of our (artificial) bird is dependent upon (1) the information it receives from the environment (through it's two sensory "detectors"), and (2) the threshold at which the MCP neuron is set. The detectors can be thought of as "looking for" particular properties. The combined input to the neuron (from the detectors) will be larger or smaller depending upon how many of the "looked-for" properties are detected by the sensors. Whether or not the total input from the sensors will be a number high enough to cause the neuron to "fire" (i.e, to give an output of "1" instead of "0") depends upon how high the threshold is.

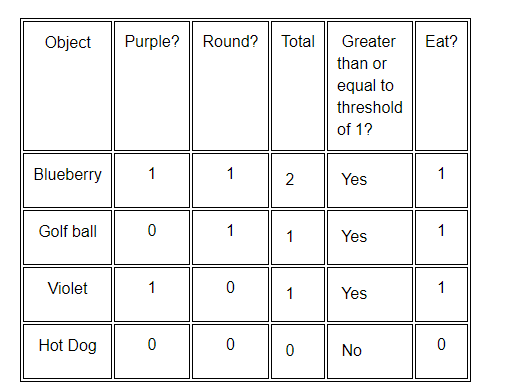
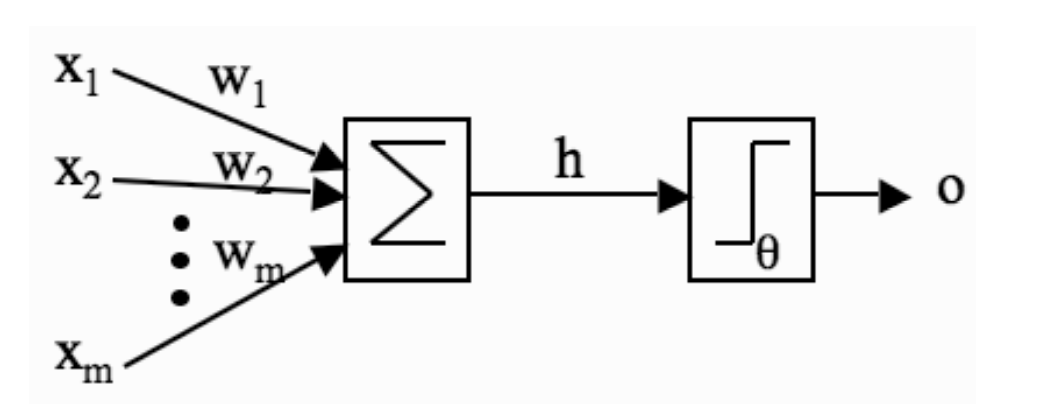


FIGURE :-A picture of McCulloch and Pitts’ mathematical model of a neuron. The inputs xi are multiplied by the weights wi , and the neurons sum their values. If this sum is greater than the threshold θ then the neuron fires; otherwise it does not.

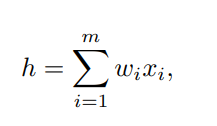


(1) **a set of weighted inputs** wi that correspond to the synapses

(2) **an adder** that sums the input signals (equivalent to the membrane of the cell that collects electrical charge)

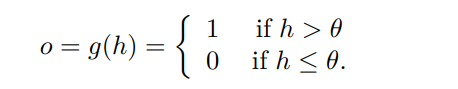
(3) **an activation function** (initially a threshold function) that decides whether the neuron fires (‘spikes’) for the current inputs.

A picture of their model is given in Figure 3.1, and we’ll use the picture to write down a mathematical description. On the left of the picture are a set of input nodes (labeled x1, x2, . . . xm). These are given some values, and as an example we’ll assume that there are three inputs, with x1 = 1, x2 = 0, x3 = 0.5. In real neurons those inputs come from the outputs of other neurons. So the 0 means that a neuron didn’t fire, the 1 means it did, and the 0.5 has no biological meaning, but never mind. (Actually, this isn’t quite fair, but it’s a long story and not very relevant.) Each of these other neuronal firings flowed along a synapse to arrive at our neuron, and those synapses have strengths, called weights. The strength of the synapse affects the strength of the signal, so we multiply the input by the weight of the synapse (so we get x1 × w1 and x2 × w2, etc.). Now when all of these signals arrive into our neuron, it adds them up to see if there is enough strength to make it fire. We’ll write that as



which just means sum (add up) all the inputs multiplied by their synaptic weights. I’ve assumed that there are m of them, where m = 3 in the example. If the synaptic weights are w1 = 1, w2 = −0.5, w3 = −1, then the inputs to our model neuron are h = 1 × 1 + 0 × −0.5 + 0.5 × −1 = 1 + 0 + −0.5 = 0.5. Now the neuron needs to decide if it is going to fire. For a real neuron, this is a question of whether the membrane potential is above some threshold. We’ll pick a threshold value (labeled θ), say θ = 0 as an example. Now, does our neuron fire? Well, h = 0.5 in the example, and 0.5 > 0, so the neuron does fire, and produces output 1. If the neuron did not fire, it would produce output 0.

The McCulloch and Pitts neuron is a binary threshold device. It sums up the inputs (multiplied by the synaptic strengths or weights) and either fires (produces output 1) or does not fire (produces output 0) depending on whether the input is above some threshold. We can write the second half of the work of the neuron(The First One is to add the inputs with the weighted synapses ), the decision about whether or not to fire (which is known as an activation function), as:



This is a very simple model, but we are going to use these neurons, or very simple variations on them using slightly different activation functions (that is, we’ll replace the threshold function with something else) for most of our study of neural networks. In fact, these neurons might look simple, but as we shall see, a network of such neurons can perform any computation that a normal computer can, provided that the weights wi are chosen correctly. So one of the main things we are going to talk about for the next few chapters is methods of setting these weights.

### Excitatory and inhibitory signals

### So far we have only considered signals coming from the bird's receivers that are added to the other signals coming from the other receivers. These types of signals are called ****excitatory**** because they excite the neuron toward possibly sending its own signal. The more excitatory signals a neuron receives, the closer the total will be to the neuron's threshold, and so the closer the neuron will be to sending its signal. So as the neuron receives more and more excitatory signals, it gets more and more excited, until the threshold is reached, and the neuron sends out its own signal. But there is another kind of signal that has the opposite effect on a neuron. These other signals are called ****inhibitory**** signals, and they have the effect of inhibiting the neuron from sending a signal. When a neuron receives an inhibitory signal, it becomes less excited, and so it takes more excitatory signals to reach the neuron's threshold. In effect, inhibitory signals subtract from the total of the excitatory signals, making the neuron more relaxed, and moving the neuron away from its threshold.

### MCP neurons with an inhibitory signal

Now let's look at an example of an MCP neuron with an inhibitory signal. Let's consider a particular type of bird, say a robin. Now the robin, which has red feathers on its breast, is safe around any red objects, including red creatures such as a cardinal. Suppose our robin's brain has a neuron with two receivers connected to the robin's eyes. Normally our robin will flee from any other creature it sees. If the robin sees another creature, an excitatory signal will be sent to the first receiver, which will try to cause the bird to flee. So the first receiver is a creature detector, and it excites our bird to fleeing. However, if the creature that the robin sees has red on it, an inhibitory signal will be sent to the second receiver, which will prevent the bird from fleeing. So the second receiver is a red detector, and it inhibits our bird from fleeing.

Suppose our robin sees a black cat. What would happen? The creature detector would send an excitatory signal to the neuron, and the red detector would send no signal. So the bird would flee.

Suppose our robin sees a cardinal. The creature detector would send an excitatory signal to the neuron, and the red detector would send an inhibitory signal. So the bird would not flee, because the inhibitory signal would "cancel" the excitatory signal.

Here is a table that summarizes how this MCP neuron with the excitatory and inhibitory signals would work in several cases.



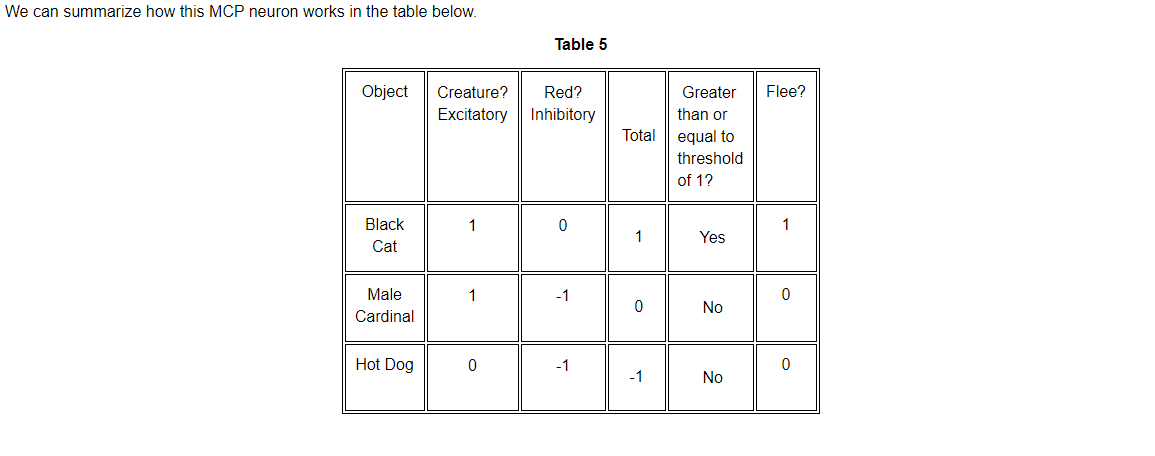
Now we'll see how these new ideas of excitatory and inhibitory signals work when the MCP neuron compares these signals to its threshold. As before, we'll use a 1 if an **excitatory** signal is sent, and a 0 if no excitatory signal is sent. But now we'll use a -1 when an **inhibitory** signal is sent, and a 0 if no inhibitory signal is sent. Because we are using a -1 for an inhibitory signal, when we add an inhibitory signal to the total of all signals received, the effect of the inhibitory signal on the total is to subtract a 1. (Recall that adding a -1 is the same as subtracting a 1.) So when an MCP neuron computes the total effect of its signals, it will add a 1 to the total for each of the excitatory signals and add a -1 to the total for each of its inhibitory signals. If the total of excitatory signals and inhibitory signals is greater than or equal to the threshold, then the MCP neuron will send a 1. If the total of excitatory signals and inhibitory signals is less than the threshold, then the MCP neuron will send a 0.

Let's look at an example. Suppose we have an MCP neuron connected to a creature detector that sends an excitatory signal and a red detector that sends an inhibitory signal. Let's also suppose the threshold is 1. We could have chosen another number for the threshold. Now for each object in Table 4, we can compute the total of the signals by adding a 1 for each excitatory signal and a -1 for each inhibitory signal. Then we compare the total to the threshold.

If the robin sees a black cat, then the creature detector, which is excitatory, sends a 1, because the cat is a creature. The red detector, which is inhibitory, sends a 0 , because the cat is not red. Because there is one excitatory signal and no inhibitory signal, the total is 1 + 0 = 1 We compare this total of 1 to the threshold. Because the total of 1 is equal to the threshold of 1, the MCP neuron will send a 1, and so the robin will flee.

If the robin sees a male cardinal, then the creature detector, which is excitatory, sends a 1, because the cardinal is a creature. The red detector, which is inhibitory, sends a -1, because the cardinal is red. Because there is one excitatory signal and one inhibitory signal, the total is 1 + -1 = 0. We compare this total of 0 to the threshold. Because 0 is less than the threshold of 1, the MCP neuron will send a 0, and so the robin will not flee.

If the robin sees a hot dog, then the creature detector, which is excitatory, sends a 0, because the hot dog is not a creature. The red detector, which is inhibitory, sends a -1, because the hot dog is red. Because there is no excitatory signal and one inhibitory signal, the total is 0 + -1 = -1. We compare this total of -1 to the threshold. Because -1 is less than the threshold of 1 , the MCP neuron will send a 0, and so the robin will not flee.



### Limitations of MCP neurons

### We now have two ways of adjusting an MCP neuron. We can make signals either excitatory or inhibitory, and we can change the threshold. By making different choices for these adjustments, we can make MCP neurons that produce a variety of results. For example, suppose we wanted to construct an MCP neuron that signals a bird to eat objects that are not both purple and round. In other words, this is an MCP neuron that does the exact opposite of what we did in our first example of an MCP neuron. ([See Table 2.](https://mind.ilstu.edu/curriculum/mcp_neurons/mcp_neuron1.html#table2)) Here's what we could do. We could make both the purple detector and the roundness detector inhibitory. Then we could set the threshold to 0.

Now if the object is a blueberry, there are two inhibitory signals sent. So the total is -1 + (-1)= -2 . When we compare -2 to the threshold of 0, we see that -2 is less than 0, so the MCP neuron will send a 0, and the bird will not eat the blueberry.

The rest of the results are summarized in the following table.

### 

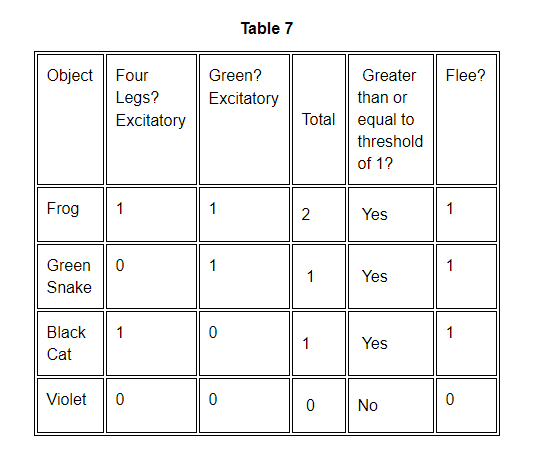
As you can see, we can indeed make an MCP neuron that does the exact opposite of the first MCP neuron that we considered. We would now like to see what other results we can produce with an MCP neuron by using various combinations of excitatory and inhibitory signals with various thresholds. For now we will only consider MCP neurons that receive two signals. Is there any limit to the type of MCP neurons we can make?

There are some limits. With only two signals, there are only three possible combinations of excitatory and inhibitory signals: both signals excitatory, both signals inhibitory, and one excitatory signal with one inhibitory signal. We've seen all of these combinations in previous examples.

Of course, the number of possible thresholds is infinite. But there are only a few types of results that an MCP neuron can produce, because it only sends out a 0 or 1. MCP neurons are also limited by the number of signals they receive. If we ignore the types of detectors and just concentrate on whether or not the detector sends a signal, we can see there are only four combinations of two signals: both detectors send a signal, the first sends a signal and the second detector does not , the second detector sends a signal and the first detector does not, and neither detector sends a signal. As a consequence, it can be shown that there are sixteen possible ways(2^4) to produce a 0 or 1 from each of the four pairs of signals. So there are at most sixteen possible types of MCP neurons with two receivers. We'll now consider a particular one of these sixteen types.

### The exclusive or

Let's consider a bird with two detectors connected to a neuron. The first detector will send a signal if the object is a creature with four legs, and the second detector sends a signal if the object is green. We want to make an MCP neuron that will signal the bird to flee if the object is either four-legged or green, but not both. We'll start by trying two excitatory signals with a threshold of 1. We'll consider four objects: a frog, a green snake, a black cat, and a violet. Notice that these objects give us the four possible combinations of 0 and 1 for each pair of signals. Also, because all the signals are excitatory, the total is just the sum of the two signals. This will produce the following results.



hile this is close to what we want, it's not perfect. The bird will flee from an object that is four-legged, such as the cat, or from an object that is green, such as the snake. But the bird will also flee from any object that is both four-legged and green, which we don't want. So this combination of signals and threshold doesn't work. In fact, what we are trying to do is make an MCP neuron that produces a 1 if either signal is a 1, but produces a 0 if both signals are either 0 or 1.

|  |
| --- |
| **Exercise 9**. Make a table like Table 7 for an MCP neuron with one excitatory signal and one inibitory signal and a threshold of 0. Does this MCP neuron produce a 1 if either signal is a 1, but produces a 0 if both signals are either 0 or 1? |

The particular type of MCP neuron we are trying to make is called an "exclusive or" MCP neuron, because it sends a 1 if it receives a 1 from one signal or the other, but not both. An "exclusive or" would have 0,1,1,0 in the last column of Table 7. There is also an "inclusive or" which sends a 1 if it receives a 1 from either signal or both. An "inclusive or" would have 1,1,1,0 in the last column of Table 7. In fact, we have already made an "inclusive or" MCP neuron in [**Table 3**](https://mind.ilstu.edu/curriculum/mcp_neurons/mcp_neuron1.html#table3).

If we try the various combinations of excitatory and inhibitory signals with several thresholds, we will begin to suspect that it is impossible to build an MCP neuron that produces an "exclusive or." In fact, it can be proved mathematically that no combination of signals and thresholds can produce the "exclusive or." So there is a fundamental limitation on the type of results a single MCP neuron can produce.

In order to produce more complicated results like the "exclusive or" we must start connecting MCP neurons together to form neural networks. It turns out that the "exclusive or" can be made by using three MCP neurons each with two receivers. Two of the MCP neurons each have one receiver attached to the first detector and the other receiver attached to the second detector. So the same signal from each detector is sent to both MCP neurons simultaneously. Each of these MCP neurons then send their signal to one of the receivers of the third MCP neuron. If the various signals and thresholds are chosen properly, this network will accept two signals and send out a signal that produces an "exclusive or." In other words, we can produce a neural network of three MCP neurons that produces an "exclusive or."

Another way to increase the possible results from an MCP neuron is to use more receivers. We could look at MCP neurons with three or more receivers. We could then link these MCP neurons together in a neural network. By doing just this, McCulloch and Pitts showed there is a neural network of MCP neurons that produces whatever combination of 1's and 0's we would like from the possible signals it receives. So while there is a fundamental limitation on a single MCP neuron, this limitation can be overcome by connecting the single MCP neurons together in a neural network. This also shows how very complicated results can be obtained by connecting together a large number of very simple parts (the MCP neurons). Perhaps this explains how our brain can do amazing things, even though it is a collection of a huge number of basic cells (real neurons) that are connected to a very large number of other basic cells.