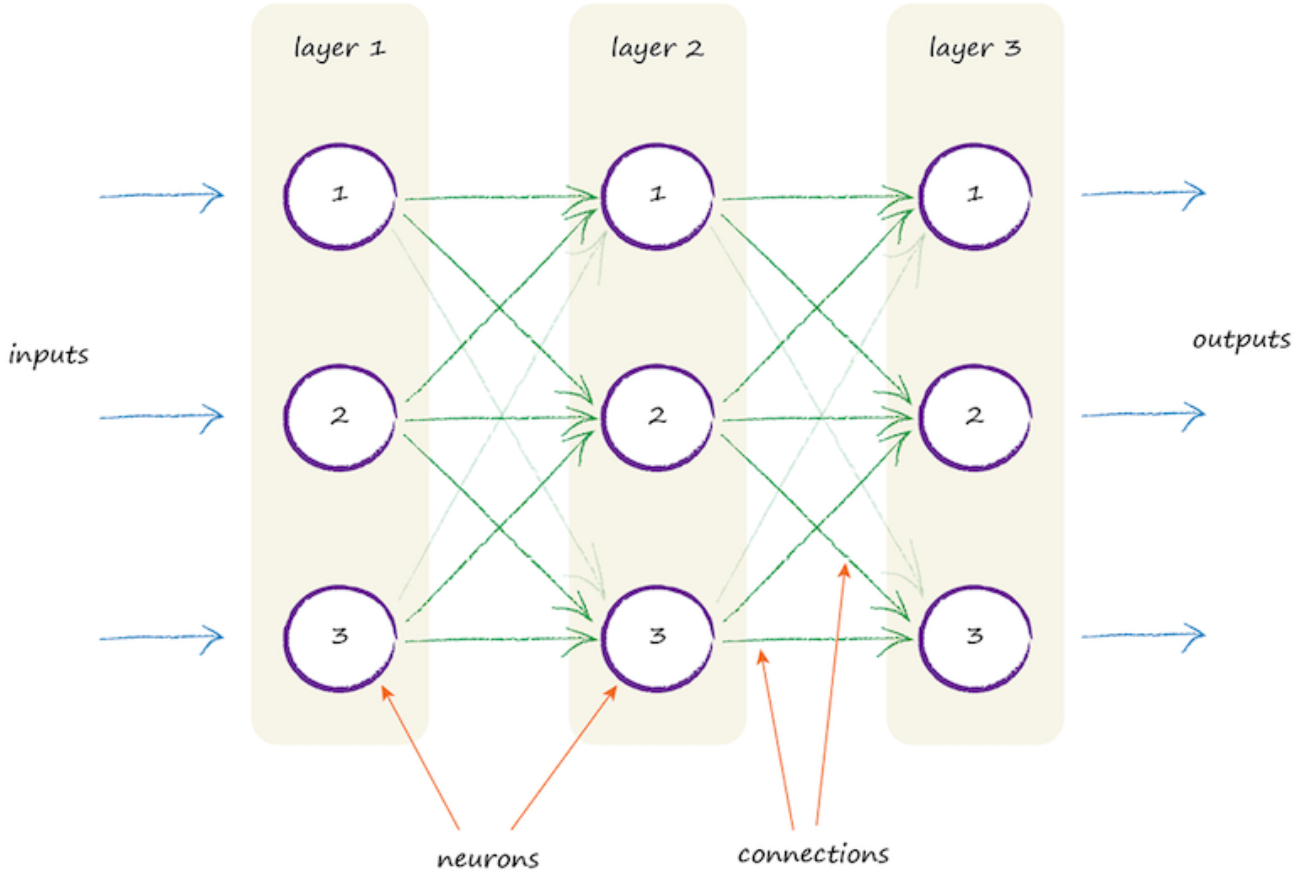
Replicating Neuron to an Artificial Model

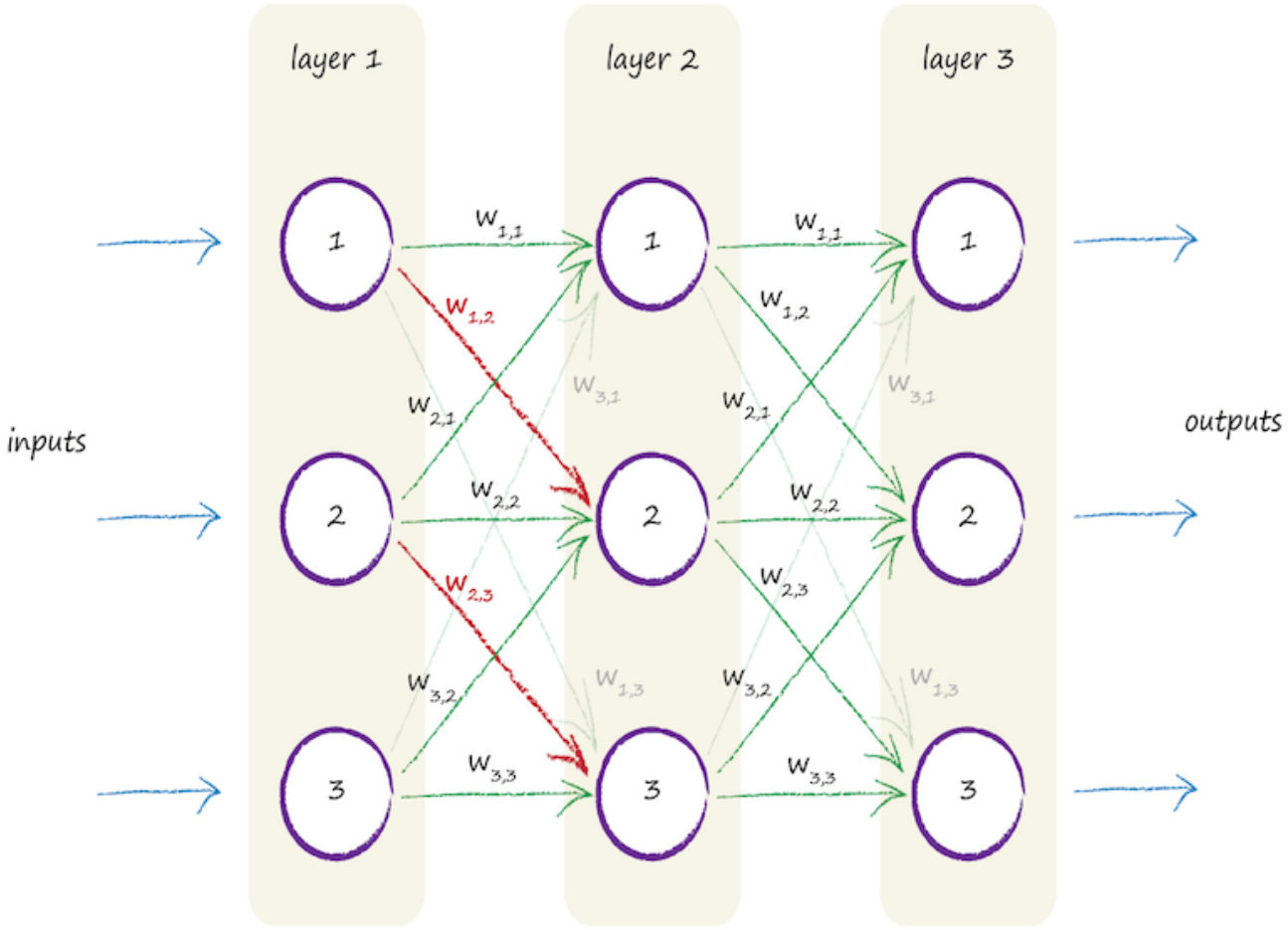
In this lesson, we will try to build our own artificial neuron with three layers, three inputs, and three outputs. We will also look at some of the basic terms that we are going to use in the further lessons.

> One way to replicate is to have layers of neurons, with each connected to every other one in the preceding and subsequent layer. The following diagram illustrates this idea:

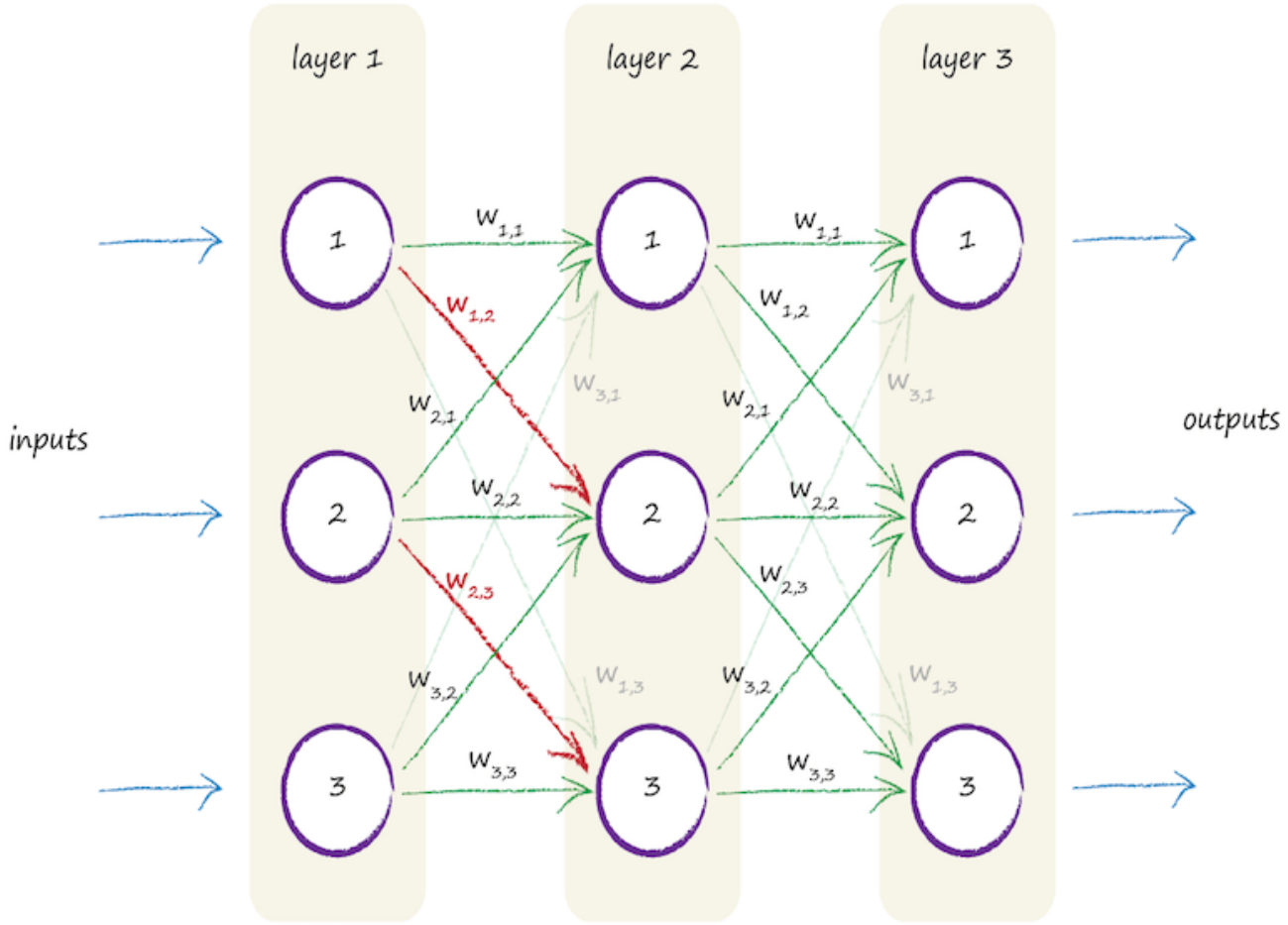


> You can see the three layers, each with three artificial neurons, or *nodes*. You can also see each node connected to every other node in the preceding and next layers. That's great! But what part of this cool looking architecture does the learning? What do we adjust in response to training examples? Is there a parameter that we can refine like the slope of the linear classifiers we looked at earlier?

The most obvious thing is to adjust the strength of the connections between nodes. Within a node, we could have adjusted the summation of the inputs, or we could have adjusted the shape of the sigmoid threshold function, but that's more complicated than simply adjusting the strength of the connections between the nodes. If the simpler approach works, let's stick with it! The following diagram again shows the connected nodes, but this time a *weight* is shown associated with each connection. A low weight will de-emphasize a signal, and a high weight will amplify it.



It is worth explaining the funny little numbers next to the weight symbols. The weight $w_{2,3}$ is simply the weight associated with the signal that passed between node 2 in a layer to node 3 in the next layer. So $w_{1,2}$ is the weight that diminishes or amplifies the signal between node 1 and node 2 in the next layer. To illustrate the idea, the following diagram shows these two connections between the first and second layer highlighted.



You might reasonably challenge this design and ask yourself why each node should connect to every other node in the previous and next layer. They don't have to, and you could connect them in all sorts of creative ways. We don't because the uniformity of this full connectivity is actually easier to encode as computer instructions, and because there shouldn't be any big harm in having a few more connections than the absolute minimum that might be needed for solving a specific task. The learning process will de-emphasize those few extra connections if they aren't actually needed.

What do we mean by this? It means that as the network learns to improve its outputs by refining the link weights inside the network, some weights become zero or close to zero. Zero or almost zero, weights means those links don't contribute to the network because signals don't pass. A zero weight means the signals are multiplied by zero, which results in zero, so the link is effectively broken.