Calculating Neural Network Output

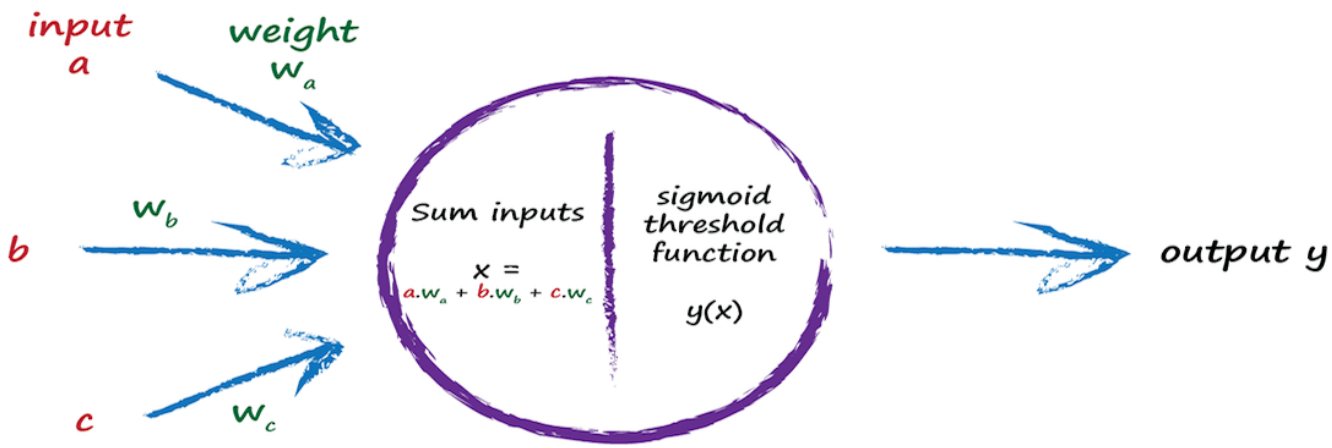
In this lesson, we will do layer-by-layer calculations to see how output is generated from the initial inputs.

Now let's do the calculations!

The first layer of nodes is the input layer, and it doesn't do anything other than representing the input signals. That is, the input nodes don't apply an activation function to the input. There is not really a fantastic reason for this other than that's how history took its course. The first layer of neural networks is the input layer and all that layer does is represent the inputs - that's it. The first input layer 1 was easy - no calculations to be done there. Next is the second layer where we do need to do some calculations. For each node in this layer, we need to work out the combined input. Remember that sigmoid function,

$$y = \frac{1}{1 + e^{-x}}$$

Well, the x in that function is the combined input into a node. That combination was the raw outputs from the connected nodes in the previous layer but moderated by the link weights. The following diagram is like the one we saw previously but now includes the need to moderate the incoming signals with the link weights.



So let's first focus on node 1 in the layer 2. Both nodes in the first input layer are connected to it. Those input nodes have raw values of 1.0 and 0.5. The

link from the first node has a weight of 0.9 associated with it. The link from the second has a weight of 0.3. So the combined moderated input is:

$x = (output\ from\ first\ node * link\ weight) + (output\ from\ second\ node * link\ weight)$

$$x = (1.0 * 0.9) + (0.5 * 0.3)$$

$$x = 0.9 + 0.15$$

$$x = 1.05$$

If we didn't moderate the signal, we would have a very simple addition of the signals $1.0 + 0.5$, but we don't want that. It is the weights that do the learning in neural networks as they are iteratively refined to give better and better results. So, we've now got $x = 1.05$ for the combined moderated input into the first node of the second layer. We can now, finally, calculate that node's output using the activation function $y = \frac{1}{1+e^{-x}}$. Feel free to use a calculator to do this. The answer is $y = 1/(1 + 0.3499) = 1/1.3499$. So, $y = 0.7408$.

That's great work! We now have an actual output from one of the network's two output nodes. Let's do the calculation again with the remaining node which is node 2 in the second layer. The combined moderated input *x* is:

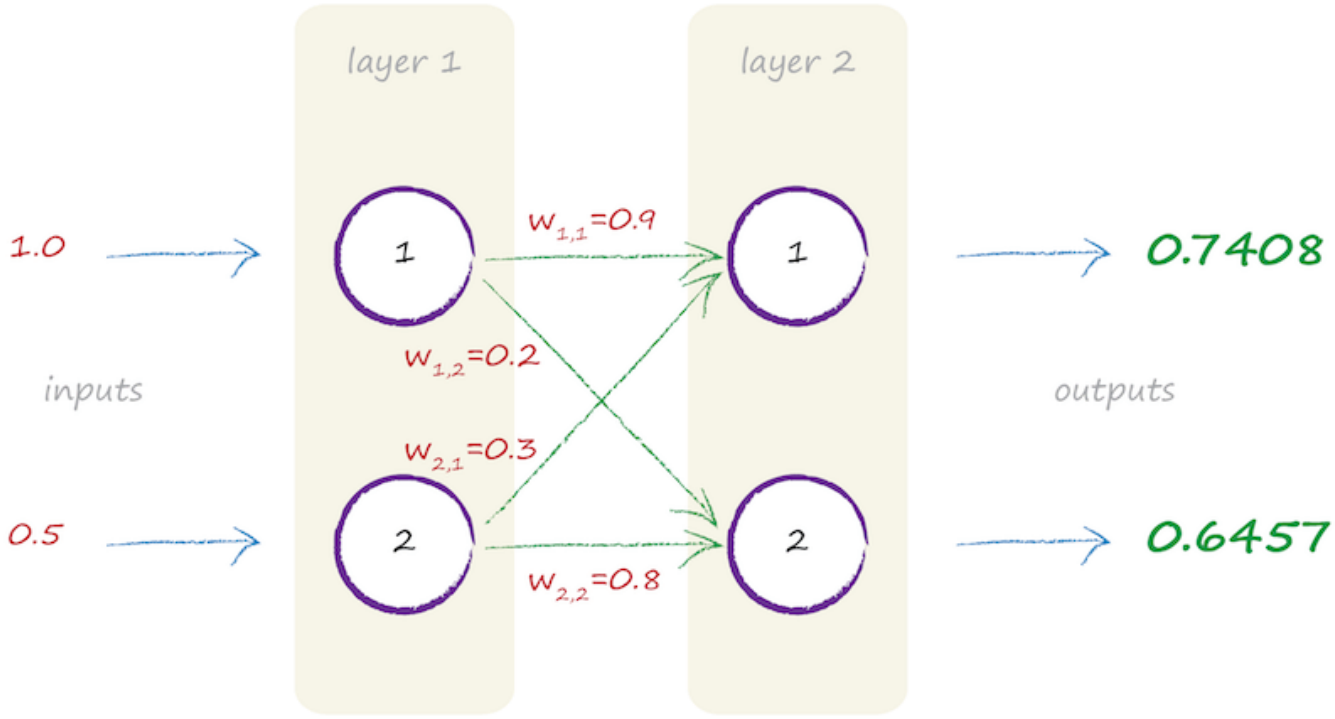
$x = (output\ from\ first\ node * link\ weight) + (output\ from\ second\ node * link\ weight)$

$$x = (1.0 * 0.2) + (0.5 * 0.8)$$

$$x = 0.2 + 0.4$$

$$x = 0.6$$

So now we have x, we can calculate the node's output using the sigmoid activation function $y = 1/(1 + 0.5488) = 1/(1.5488)$. So $y = 0.6457$. The following diagram shows the network's outputs we've just calculated:



That was a fair bit of work just to get two outputs from a very simplified network. I wouldn't want to do the calculations for a larger network by hand at all! Luckily computers are perfect for doing lots of calculations quickly and without getting bored. Even so, I don't want to write out computer instructions for doing the calculation for a network with more than 2 layers, and with maybe 4, 8 or even 100s of nodes in each layer. Even writing out the instructions would get boring and I would very likely make mistakes writing all those instructions for all those nodes and all those layers, never mind actually doing the calculations.

Luckily, mathematics helps us be extremely concise in writing down the calculations needed to work out the output from a neural network, even one with many more layers and nodes. This conciseness is not just good for us human readers, but also great for computers too because the instructions are much shorter and the execution is much more efficient too. This concise approach uses *matrices*, which we'll look at next.