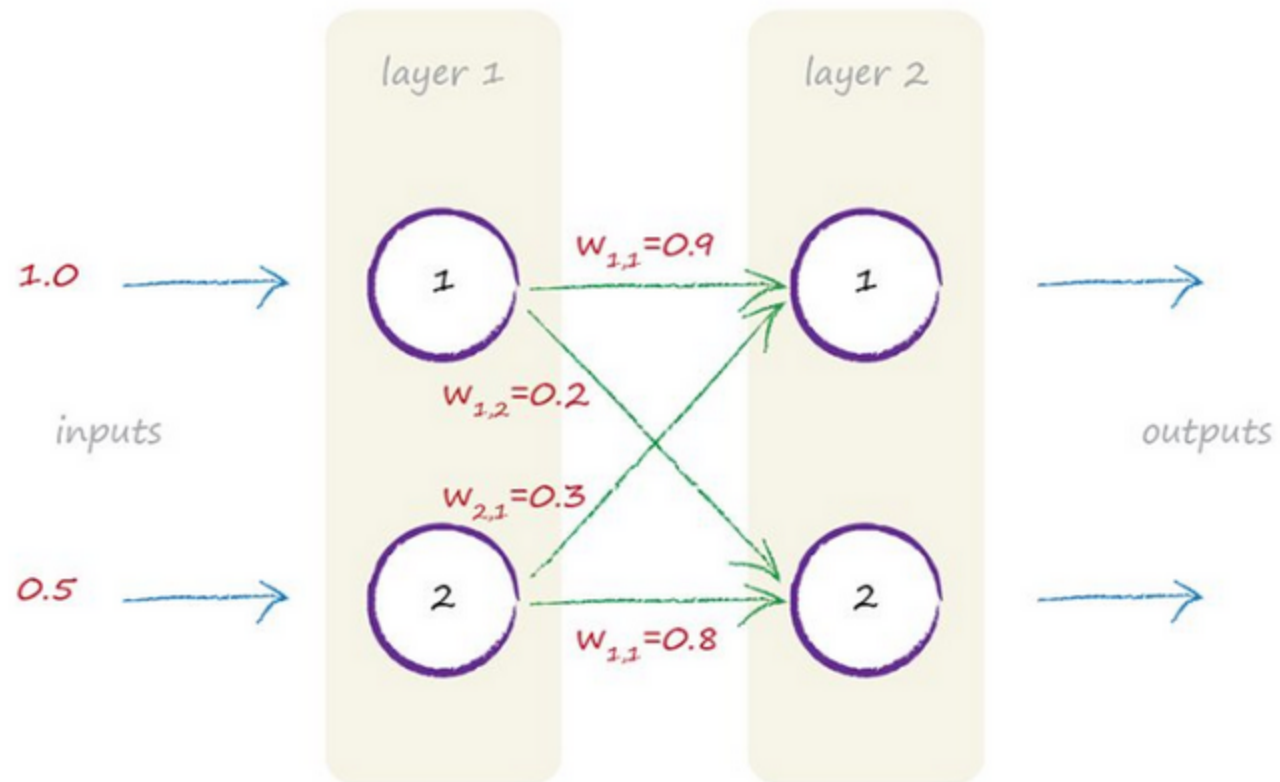


Matrix Multiplication is Useful



Let take this example and manually calculate the result of each layer.

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

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

$$x = 0.9 + 0.15$$

$$x = 1.05$$

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

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

$$x = 0.2 + 0.4$$

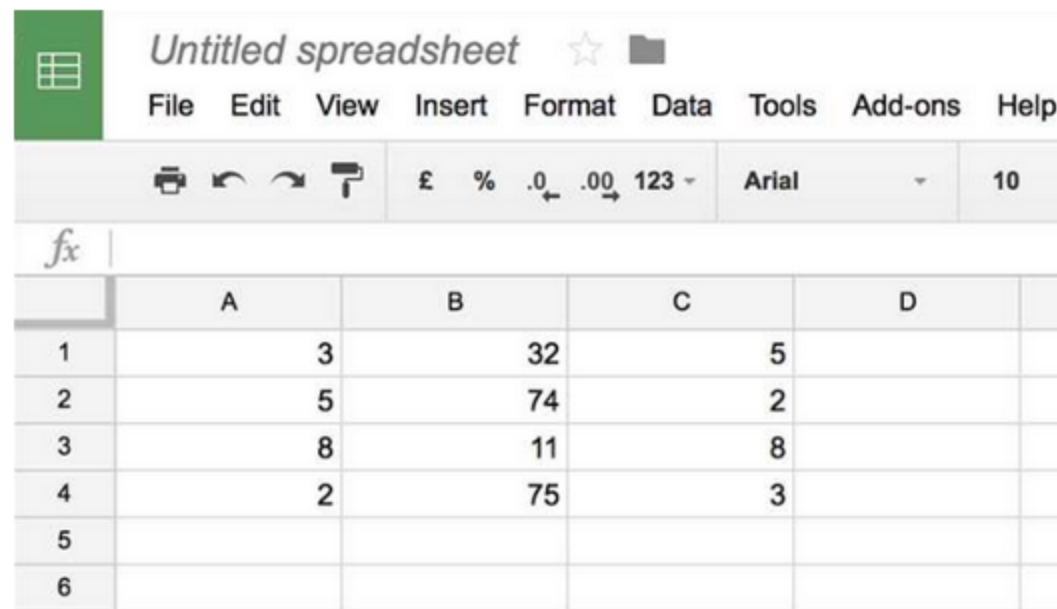
$$x = 0.6$$

That was enough work, but imagine doing the same for a network with 5 layers and 100 nodes in each? Just writing out all the necessary calculations would be a huge task ... all those combinations of combining signals, multiplied by the right weights, applying the sigmoid activation function, for each node, for each layer ... argh!

So how can matrices help? Well, they help us in two ways. First, they allow us to compress writing all those calculations into a very simple short form. That's great for us humans, because we don't like doing a lot of work because it's boring, and we're prone to errors anyway. The second benefit is that many computer programming languages understand working with matrices, and because the real work is repetitive, they can recognise that and do it very quickly and efficiently.

A matrix is just a table, rectangular grid of numbers. There nothing complex to that.

All of us have used Excel Spreadsheets, this is what a matrix is in it.



	A	B	C	D
1	3	32	5	
2	5	74	2	
3	8	11	8	
4	2	75	3	
5				
6				

Let take an example of a 2 by 3 matrix. Remember its rows first then columns so its not 3 by 2

Let take an example of a 2 by 3 matrix. Remember its rows first then columns so its not 3 by 2.

$$\begin{pmatrix} 23 & 43 & 22 \\ 43 & 12 & 54 \end{pmatrix}$$

Let take a simple example of two 2 by matrices being multiplied with each other.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} (1*5) + (2*7) & (1*6) + (2*8) \\ (3*5) + (4*7) & (3*6) + (4*8) \end{pmatrix}$$
$$= \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

The following shows the rules used by using variables instead of numbers.

$$\begin{pmatrix} a & b & \dots \\ c & d & \dots \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} (a*e) + (b*g) + \dots & (a*f) + (b*h) + \dots \\ (c*e) + (d*g) + \dots & (c*f) + (d*h) + \dots \end{pmatrix}$$

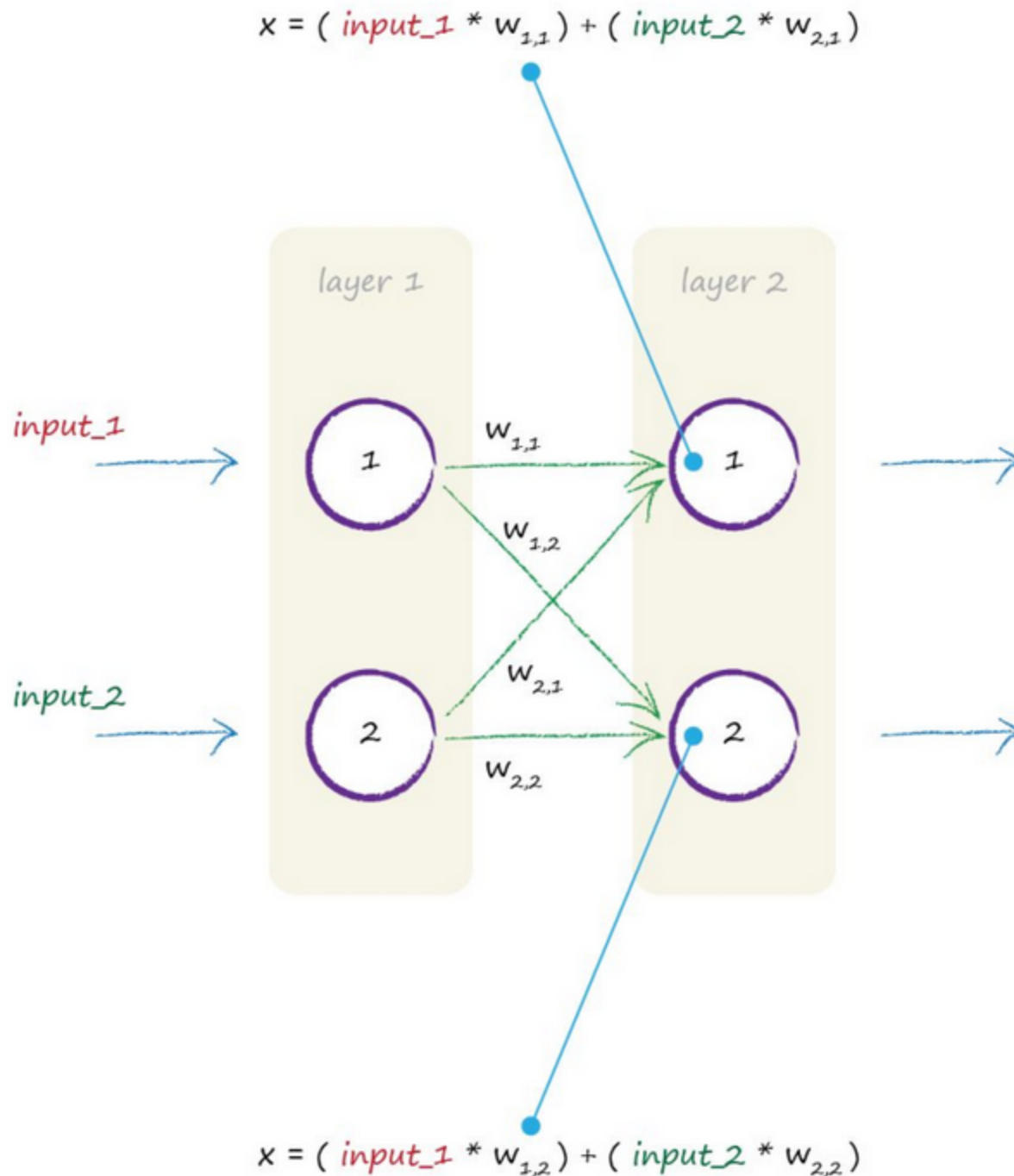
$$= \begin{pmatrix} ae+bg+\dots & af+bh+\dots \\ ce+dg+\dots & cf+dh+\dots \end{pmatrix}$$

Now let's write the matrices for our neural network. It's far more meaningful to write it like this

$$\begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix} \begin{pmatrix} \text{input}_1 \\ \text{input}_2 \end{pmatrix} = \begin{pmatrix} (\text{input}_1 * w_{1,1}) + (\text{input}_2 * w_{2,1}) \\ (\text{input}_1 * w_{1,2}) + (\text{input}_2 * w_{2,2}) \end{pmatrix}$$

The first matrix contains the weights between nodes of two layers. The second matrix contains the signals of the first input layer. The answer we get by multiplying these two matrices is the combined moderated signal into the nodes of the second layer. Look carefully, and you'll see this. The first node has the first input_1 moderated by the weight $w_{1,1}$, added to the second input_2 moderated by the weight $w_{2,1}$. These are the values of x before the sigmoid activation function is applied.

The following diagram shows this even more clearly.



That is really useful. Why? Because we can express all the calculations that go into working out the combined moderated signal, x , into each node of the second layer using matrix multiplication. And this can be expressed as concisely as:

$$X = WI$$

That is, W is the matrix of weights, I is the matrix of inputs, and X is the resultant matrix of combined moderated signals into layer 2. Matrices are often written in bold to show that they are in fact matrices and don't just represent single numbers.

We now don't need to care so much about how many nodes there are in each layer. If we have more nodes, the matrices will just be bigger. But we don't need to write anything longer or larger. We can simply write $W \cdot I$ even if I has 2 elements or 200 elements! Now, if a computer programming language can understand matrix notation, it can do all the hard work of many calculations to work out the $X = W \cdot I$, without us having to give it all the individual instructions for each node in each layer.

What about the activation function? That's easy and doesn't need matrix multiplication. All we need to do is apply the sigmoid function $y = \text{sigmoid}(x)$ to each individual element of the matrix X .

This sounds too simple, but it is correct because we're not combining signals from different nodes here, we've already done that and the answers are in X . As we saw earlier, the activation function simply applies a threshold and squishes the response to be more like that seen in

$$O = \text{sigmoid}(X)$$

The O written in Bold is a matrix, that contains all the output of the final layer. This is fantastic! A little bit of effort to understand matrix multiplication has given us a powerful tool for implementing neural networks without lots of effort from us.