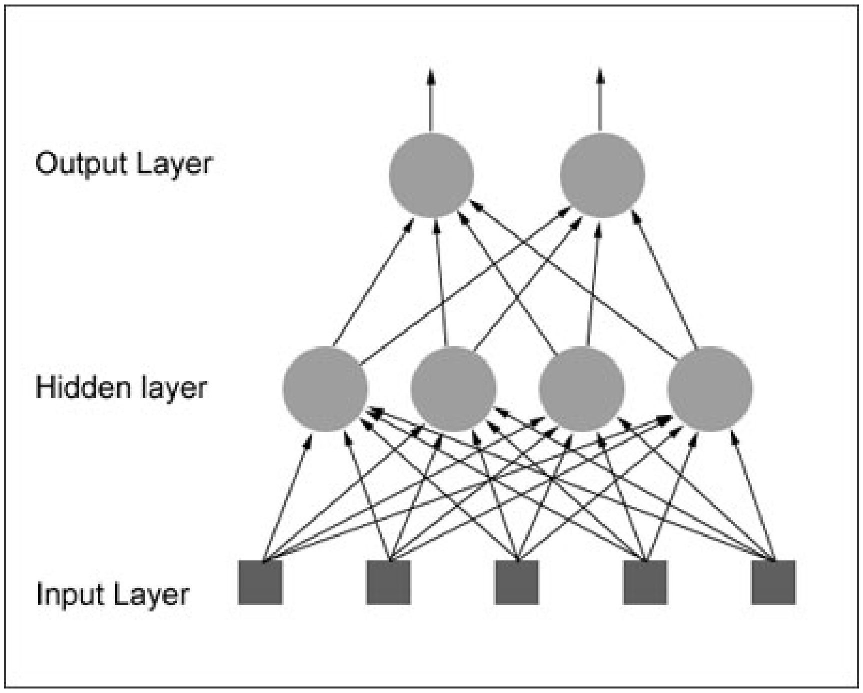
a i - j u n k i e

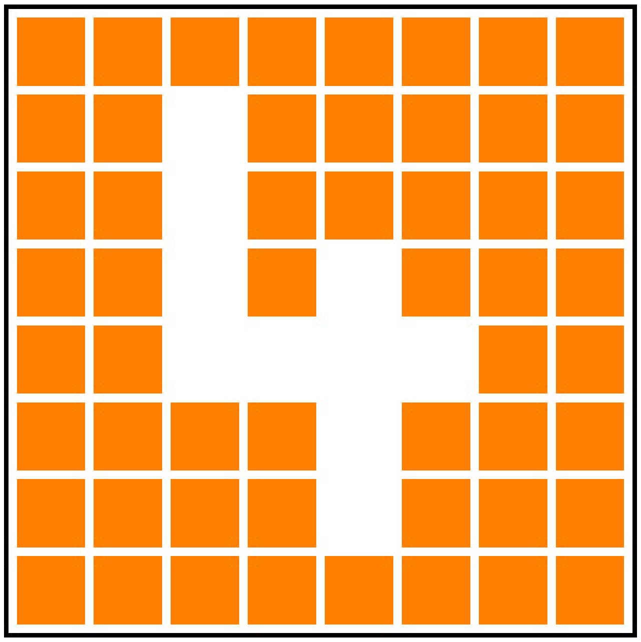
I understand all that but how do you actually *use* an artificial neuron?

Well, we have to link several of these neurons up in some way. One way of doing this is by organising the neurons into a design called a *feedforward network*. It gets its name from the way the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network. This is what a very simple feedforward network looks like:



Each input is sent to every neuron in the hidden layer and then each hidden layer's neuron's output is connected to every neuron in the next layer. There can be any number of hidden layers within a feedforward network but one is usually enough to suffice for most problems you will tackle. Also the number of neurons I've chosen for the above diagram was completely arbitrary. There can be any number of neurons in each layer, it all depends on the problem. By now you may be feeling a little dazed by all this information so I think the best thing I can do at this point would be to give you a real world example of how a neural net can be used in the hope that I can get your very own brain's neurons firing!

You probably know already that a popular use for neural nets is character recognition. So let's design a neural network that will detect the number '4'. Given a panel made up of a grid of lights which can be either on or off, we want our neural net to let us know whenever it thinks it sees the character '4'. The panel is eight cells square and looks like this:



We would like to design a neural net that will accept the state of the panel as an input and will output either a 1 or zero. A 1 to indicate that it thinks the character ‘4’ is being displayed and 0 if it thinks it's not being displayed. Therefore the neural net will have 64 inputs, each one representing a particular cell in the panel and a hidden layer consisting of a number of neurons (more on this later) all feeding their output into just one neuron in the output layer. I hope you can picture this in your head because the thought of drawing all those little circles and lines for you is not a happy one <smile>.

Once the neural network has been created it needs to be trained. One way of doing this is initialize the neural net with random weights and then feed it a series of inputs which represent, in this example, the different panel configurations. For each configuration we check to see what its output is and adjust the weights accordingly so that whenever it sees something looking like a number 4 it outputs a 1 and for everything else it outputs a zero. This type of training is called *supervised learning* and the data we feed it is called a *training set*. There are many different ways of adjusting the weights, the most common for this type of problem is called *backpropagation*. I will not be going into backprop in this tutorial as I will be showing you a completely different way of training neural nets which requires no supervision whatsoever (and hardly any maths - woohoo!)

If you think about it, you could increase the outputs of this neural net to 10. This way the network can be trained to recognize all the digits 0 through to 9. Increase them further and it could be trained to recognize the alphabet too!

Are you starting to get a feel for neural nets now? I hope so. But even if you’re not all that will hopefully change in a moment when you start to see some code.

1 2 3 5 6 7 8 **Next** **Home**