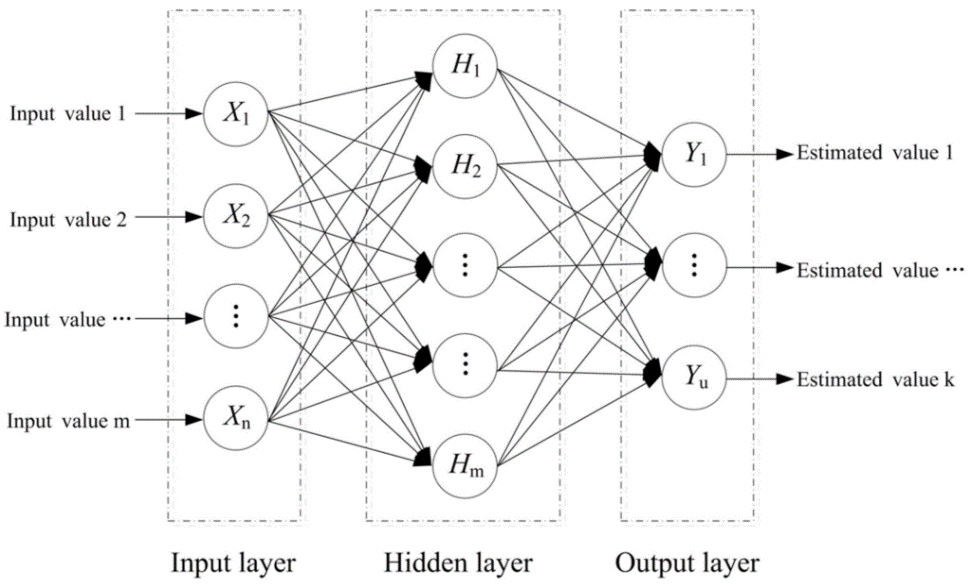
Supervised Backpropagation Artificial Neural Networks

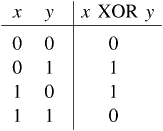
In simple terms, a Supervised Artificial Neural Network is a model, inspired by biological neural networks, with which you can “teach” a computer to approximate a function by “learning” from datasets of inputs and their corresponding outputs (these inputs and outputs are generally 1 or 0 values). In other words, given inputs and their corresponding outputs, the neural network can “learn” to approximate the function that results in those outputs from those inputs. An artificial neural network consists of neurons that are arranged in neuron layers. They can be visualised as shown in the diagram (you can have multiple hidden layers):

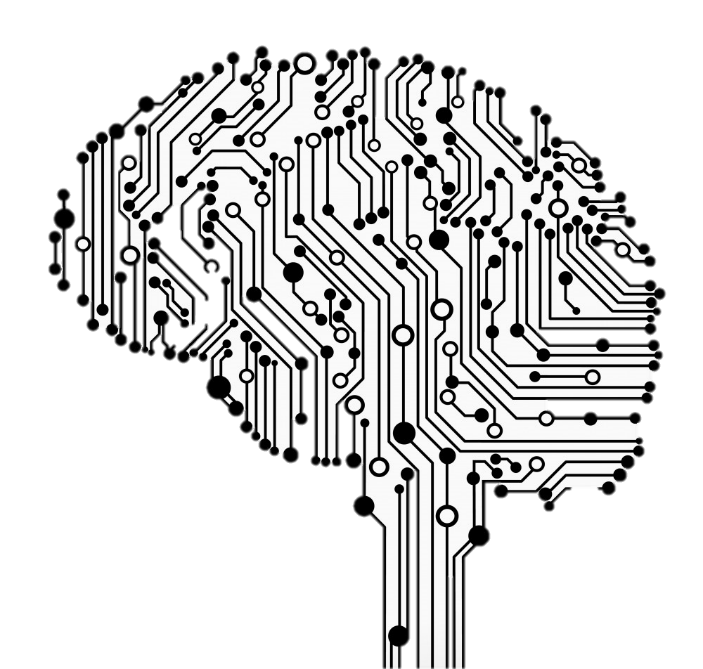
Each neuron has a number of inputs and outputs. The inputs to that neuron are modified by a weight which can reduce or increase the input value (these weights start off as small, random values – this is to increase neural network training efficiency). To calculate the output of a particular neuron, all of the modified inputs to a the neuron are summed, resulting in the internal value , which is then biased using an Activation Function (). The activation function, in my case, is the sigmoid function .

Another commonly used activation function is the threshold function .

In other words, .

This specific model of artificial neural network “learns” by updating the biases (the bias offsets the activation function) and weights of neurons in order to get closer to the desired output in learning steps known as epochs. The distance of the current solution from the optimal solution is called the cost, , which is calculated using a Cost Function, for example, the cross-entropy cost function Essentially, the aim of an epoch is to minimise the cost.

Eventually, after running through many learning epochs of many different (preferably every single combination) of inputs and their corresponding outputs, the neural network will be at a position in which all of the weights and biases are aligned in a way such that it is able to accurately approximate the outputs of any given inputs – thereby approximating a function from just its inputs and outputs. This method of learning is known as backpropagation and is but one of many artificial neural network models.

Consider the XOR gate, for example. It has 2 inputs and 1 output and performs the XOR function. We can approximate this function using a neural network by making a network of 2 input neurons, 2 hidden neurons and 1 output neuron and by training the network with the XOR truth table, for epochs ( is proportional to accuracy).

The most fascinating thing about artificial neural networks, however, is that they can be expanded to a scale limited only (*theoretically*) by your imagination. Bear in mind that you *can* input images into neural networks by assigning 1 (or more – for stuff like colour) input neuron for every pixel. In other words, using neural networks, you can very well create a program to differentiate pictures of cats from pictures of ducks or recognise fingerprints or tackle the traveling salesman problem or even read your handwriting – and that is exactly what I did. I created an advanced neural network library in C# and made a program that outputs a 6 bit representation of handwritten characters! More information on this project can be found here: <http://pixelzerg.github.io/pm/#NeuroOCR>

By Abhinav Bhandari – 9W