Optional: More Sophisticated Weights

In this lesson, we will try to refine the initializing weights. This step is optional and can be skipped while building a Neural Network but it's a very good approach to get good accuracy on your test data.

This bit is optional as it is just a simple but popular refinement to initializing weights.

If you followed the earlier discussion in part 1 of this guide about preparing data and initializing weights at the end of the first part of this guide, you would have seen that some prefer a slightly more sophisticated approach to creating the initial random weights. They sample the weights from a normal probability distribution centered around zero and with a standard deviation that is related to the number of incoming links into a node, 1/√(number of incoming links).

This is easy to do with the help of the *numpy* library. Again Google is really helpful in finding the right documentation. The `numpy.random.normal()` function described here, helps us sample a normal distribution.

The parameters are the center of the distribution, the standard deviation and the size of a *numpy* array if we want a matrix of random numbers instead of just a single number. Our updated code to initialize the weights will look like this:

```
self.wih = numpy.random.normal(0.0, pow(self.hnodes, -0.5), (self.hnodes, self.inodes))
self.who = numpy.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes))
```

You can see we've set the center of the normal distribution to 0.0. You can see the expression for the standard deviation related to the number of nodes in next layer in Python form as pow(`self.hnodes`, -0.5) which is simply raising the number of nodes to the power of -0.5. That last parameter is the shape of the *numpy* array we want.