

The Complete Neural Network Code

Here's the complete code of the Neural Network that we have built so far.

We've completed the neural network class. Here it is for reference, and you can always get it from the following link to GitHub, an online place for sharing code:

- https://github.com/makeyourownneuralnetwork/makeyourownneuralnetwork/blob/master/part2_neural_network.ipynb

```
# neural network class definition
class neuralNetwork:

    # initialise the neural network
    def __init__(self, inputnodes, hiddennodes, outputnodes, learningrate):
        # set number of nodes in each input, hidden, output layer
        self.inodes = inputnodes
        self.hnodes = hiddennodes
        self.onodes = outputnodes

        # link weight matrices, wih and who
        # weights inside the arrays are w_i_j, where link is from node i to node j in the next layer
        # w11 w21
        # w12 w22 etc
        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))

        # learning rate
        self.lr = learningrate

        # activation function is the sigmoid function
        self.activation_function = lambda x: scipy.special.expit(x)

    pass

    # train the neural network
    def train(self, inputs_list, targets_list):
        # convert inputs list to 2d array
        inputs = numpy.array(inputs_list, ndmin=2).T
        targets = numpy.array(targets_list, ndmin=2).T

        # calculate signals into hidden layer
        hidden_inputs = numpy.dot(self.wih, inputs)
        # calculate the signals emerging from hidden layer
```

```

# calculate the signals emerging from hidden layer
hidden_outputs = self.activation_function(hidden_inputs)

# calculate signals into final output layer
final_inputs = numpy.dot(self.who, hidden_outputs)
# calculate the signals emerging from final output layer
final_outputs = self.activation_function(final_inputs)

# output layer error is the (target - actual)
output_errors = targets - final_outputs
# hidden layer error is the output_errors, split by weights, recombined at hidden nodes
hidden_errors = numpy.dot(self.who.T, output_errors)

# update the weights for the links between the hidden and output layers
self.who += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)),
                                 numpy.transpose(hidden_outputs))

# update the weights for the links between the input and hidden layers
self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs * (1.0 - hidden_outputs)),
                                 numpy.transpose(inputs_list))

pass

# query the neural network
def query(self, inputs_list):
    # convert inputs list to 2d array
    inputs = numpy.array(inputs_list, ndmin=2).T

    # calculate signals into hidden layer
    hidden_inputs = numpy.dot(self.wih, inputs)
    # calculate the signals emerging from hidden layer
    hidden_outputs = self.activation_function(hidden_inputs)

    # calculate signals into final output layer
    final_inputs = numpy.dot(self.who, hidden_outputs)
    # calculate the signals emerging from final output layer
    final_outputs = self.activation_function(final_inputs)

    return final_outputs

```

There's not a lot of code here, especially if you appreciate that this code can be used to create, train and query 3-layer neural networks for almost any task. Next, we'll work on our specific task of learning to recognize numbers written by humans.