```
# python notebook for Make Your Own Neural Network
# (c) Tariq Rashid, 2016
# license is GPLv2
```

```
Import numpy
# scipy.special for the signed function expit()
Import scpicy.special

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

      #print matrix weights
      print("weights: ", self.wih, self.who)

      # 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
        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))


        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
```

In [4]:
```
# number of input, hidden and output nodes
input_nodes = 3
hidden_nodes = 300
output_nodes = 3

# learning rate is 0.3
learning_rate = 0.3

# create instance of neural network
n = neuralNetwork(input_nodes,hidden_nodes,output_nodes, learning_rate)
```

No one knows what the weights in this hidden layer mean. It's probably just error correction.

```
weights:  [[-4.20021998e-01 -1.68708106e-01  4.43144563e-01]
 [ 2.35539290e-01 -1.15728150e-01 -4.92650193e-01]
 [ 6.88755755e-01 -3.85608440e-01  4.27108021e-01]
 [-4.50177530e-01 -1.59515045e+00  9.64176843e-01]
 [-4.47529306e-02 -3.03669573e-01  1.96486657e-01]
 [ 4.24924544e-01 -1.91067138e-01  5.88321076e-01]
 [ 6.79207256e-02 -2.62896348e-01 -2.50330715e-01]
 [ 8.96895006e-01 -6.26000031e-01 -1.46246387e+00]
 [ 1.26969384e-01  6.47355275e-01  7.52656797e-01]
 [ 2.28948571e-02  8.61081804e-02  2.95430645e-01]
 [ 1.09010740e+00  7.89838058e-01 -4.38797961e-01]
 [-1.48538918e-01 -7.76146798e-01  1.94870692e-01]
 [-8.81638380e-01  5.52562986e-01 -7.27877131e-02]
 [ 6.33508458e-01 -4.64962422e-05  8.31717633e-02]
 [ 1.38304878e-01  1.03076542e+00 -4.39062121e-01]
 [-1.08964204e+00  5.67648208e-01  4.38446751e-01]
 [-3.17515932e-01  7.07908325e-01  1.22331811e+00]
 [-7.08759959e-01  5.92878151e-01 -1.46013271e+00]
 [-2.87441774e-01  7.83781478e-01  3.89814892e-01]
```

In [ ]:

In [6]:
```
n.train([0,0,0],[.10,.10,.10])
n.query([.10,.10,.10])
```

[.2,.2,.2] [.3,.3,.3]

Out[6]:
```
array([[0.0918166 ],
       [0.10061511],
       [0.09281288]])
```

[.118] [.164] [.263]
[.106] [.166] [.240]
[.124] [.195] [.295]
[.110] [.148] [.359]
[.093] [.211] [.273]
[.097] [.160] [.224]
[.154] [.231]
[.151] [.234]
[.183] [.236]

$\bar{x} = .103$, $\sigma = .012$

$\bar{x} = .170$, $\sigma = .021$

$\bar{x} = .261$, $\sigma = .043$