The below Neural Networks is a mathematical function that maps a given input to a desired output. here the NN consists of a input layer, output layer, hidden layers and a set of weights and biases.

Here the activation function used is a sigmoid function, here the loss fuction used is a sum of squares error.

Python Code:

```
import
numpy
as np

        def sigmoid(x):
            return 1.0/(1+ np.exp(-x))

        def sigmoid_derivative(x):
            return x * (1.0 - x)

        class NeuralNetwork:
            def __init__(self, x, y):
                self.input      = x
                self.weights1   = np.random.rand(self.input.shape[1],4)
                self.weights2   = np.random.rand(4,1)
                self.y          = y
                self.output     = np.zeros(self.y.shape)

            def feedforward(self):
                self.layer1 = sigmoid(np.dot(self.input, self.weights1))
                self.output = sigmoid(np.dot(self.layer1, self.weights2))

            def backprop(self):
                # application of the chain rule to find derivative of the loss function with respect to weights2 and weights1
                d_weights2 = np.dot(self.layer1.T, (2*(self.y - self.output) * sigmoid_derivative(self.output)))
                d_weights1 = np.dot(self.input.T,  (np.dot(2*(self.y - self.output) * sigmoid_derivative(self.output), self.weights2.T) *
        sigmoid_derivative(self.layer1)))



                self.weights1 += d_weights1
                self.weights2 += d_weights2


        if __name__ == "__main__":
            X = np.array([[0,0,1],
                          [0,1,1],
                          [1,0,1],
                          [1,1,1]])
            y = np.array([[0],[1],[1],[0]])
            nn = NeuralNetwork(X,y)

            for i in range(1500):
                nn.feedforward()
                nn.backprop()

            print(nn.output)
```

apply our Neural Network on an example

X1	X2	X3	Y
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

After some iterations

predictions	Y(actual)
0.023	0
0.980	1
0.970	1
0.015	0