**Function Neural Network Manual**

## MSc. Eber Jair Flores Andonegui.

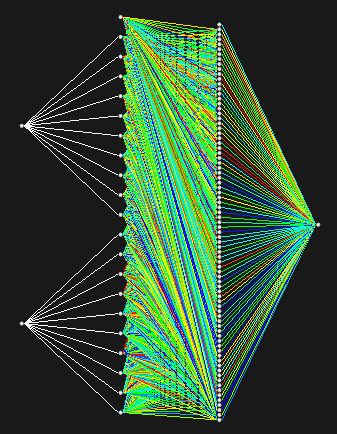
## eberjair@gmail.com

# Application objective

The objective of this application is to show a sample of simple neural network and its learning process using visualization tools. The neural network is designed to learn and imitate the behavior of any continuous two variables function in a determined domain.

# Neural Network Structure

The Neural Network has been built with the input layer, two hidden layers (20 and 80 neurons) and the output layer as shown in the Neural Network Viewer in the app:



# Hidden layers and Output layer activation functions

## First hidden layer

The first hidden layer acts a as semi input layer, its only function is to characterize somehow the values from the input layer given by **X1** and **X2** input values which must be **normalized**.

The activation function is given by:

(1)

where:

, , , … , , , , …

## Second hidden layer

The second hidden is a fully connected layer and its activation function is given by:

(2)

(3)

where:

is the weight coming from a bias unit for this layer.

is the weight for the connection between the neuron of the first hidden layer and the neuron of the second hidden layer.

## Output layer

The output layer is fully connected layer with only one neuron which gives the output value. The activation layer is given by:

(4)

(5)

where:

is the weight coming from a bias unit for this layer.

is the weight for the connection between the neuron of the second hidden layer and the neuron of the output layer.

# Neural network learning process

The method used for the learning process is supervised learning with backpropagation to adjust the weights between neurons.

The used error function is given by:

(6)

where is the expected output value.

For the backpropagation we try to minimize the error function for a set of random inputs using max descend optimization.

## Adjusting

To adjust we use the next equation for each new learning iteration:

(7)

where is the gradient factor and:

(8)

Using (6), (4) and (5) we get:

(9)

## Adjusting

To adjust we use the next equation for each new learning iteration:

(7)

where is the gradient factor and:

(8)

Using (6), (4) and (5) we get:

(9)