**Function Neural Network Design**

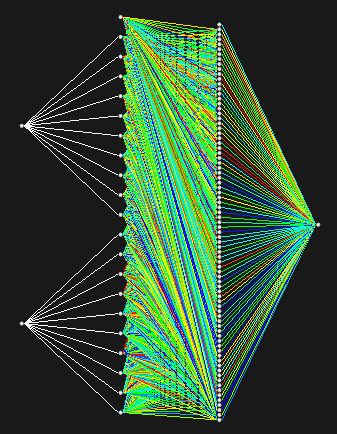
## MSc. Eber Jair Flores Andonegui.

## eberjair@gmail.com

# Neural Network Structure

This Neural Network is designed to learn to imitate any two variables continuous function for a determined domain.

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 Function Neural Network application:



# Hidden layers and Output layer activation functions

## First hidden layer

The first hidden layer acts as a 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 for this layer is given by:

(1)

where:

, , , … , , , , …

## Second hidden layer

The second hidden layer 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 a fully connected layer with only one neuron which gives the output value. The activation function for this 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 method 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 the learning iterations:

(7)

where is the [gradient factor](#_Gradient_factor_𝜂) and:

(8)

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

(9)

## Adjusting

To adjust we use the next equation for the learning iterations:

(10)

where is the [gradient factor](#_Gradient_factor_𝜂) and:

(11)

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

(12)

## Adjusting

To adjust we use the next equation for the learning iterations:

(13)

where is the [gradient factor](#_Gradient_factor_𝜂) and:

(14)

Using (6), (4), (5), (2) and (3) we get:

(15)

## Adjusting

To adjust we use the next equation for the learning iterations:

(16)

where is the [gradient factor](#_Gradient_factor_𝜂) and:

(17)

Using (6), (4), (5), (2) and (3) we get:

(18)

## Gradient factor

There are four possible values for the gradient factor given by:

,

,

or

where is the iteration of the learning process for a total of iterations