Rubén Romero Hernández **rromero@mion.elka.pw.edu.pl**

PROJECT NEURAL NETWORK

This proyect consist in desinging a multilayer perceptron for function approximation y=f(x) based on a given training set. The structure of this network will be 2 input neurons: x and 1 (bias), one hidden layer of m neurons, one output neuron. As activation function we will use the tanh(x).

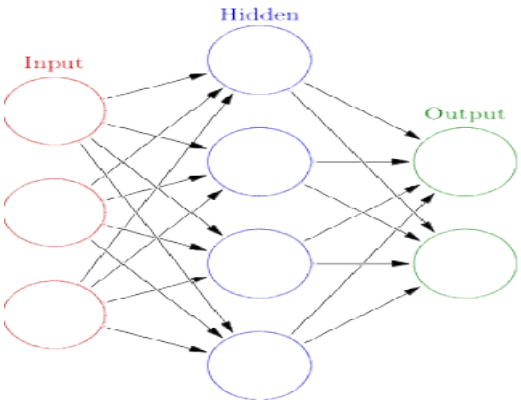
The first step of this proyect will be generating the data test. We will generate two sets of data of size 40 and 400, according to the next equation:

$$y(x) = \frac{\sin(x)}{x} + \varepsilon$$

Where $x \in (0, 15)$ and ε – random number uniformly distributed in the interval (-0.1, 0.1).

Secondly, we have to desing our neural network. To desing our neural network, we wil use the matlab application, because this application include several tools very useful for working with neural network.

The goal of our neural network is the minimization of mean square error of function approximation. To archieve this goal we should choose the correct number of hidden neurons and adjust the weight of each conection.



The structure will consist in two inputs, one hidden layer with m neurons, and one output. The input will be connected with the m hidden neurons, the hidden neuron will be connected with the output neuron, and each connection will have a specific weight. Each hidden neuron will receive the addition of each input multiply by the weight of his conection. After, in the hidden neurons each neuron will apply the activation function that will be hyperbolic tangent, tanh. Finally, the result of each hidden neuron multiply by the weight of his conection with the output neuron will be delivered to this neuron, where every result will be added to the final outcome.

¿HOW IS THIS NEURAL NETWORK GOING TO ESTIMATE THE RESULT? (TRAINING)

If we choose a random weights between [-0.15, 0.15], and try to estimate a solution, the estimation will be really bad (mse hight). These estimations don´t have any accuracy due to the neural network needs a training to adjust the weight properly, not randomly like we did.

Result without training, using set of 40 datas input.

Number of hidden network	Training set mse	Test set mse
10	Random initialization	29.8333

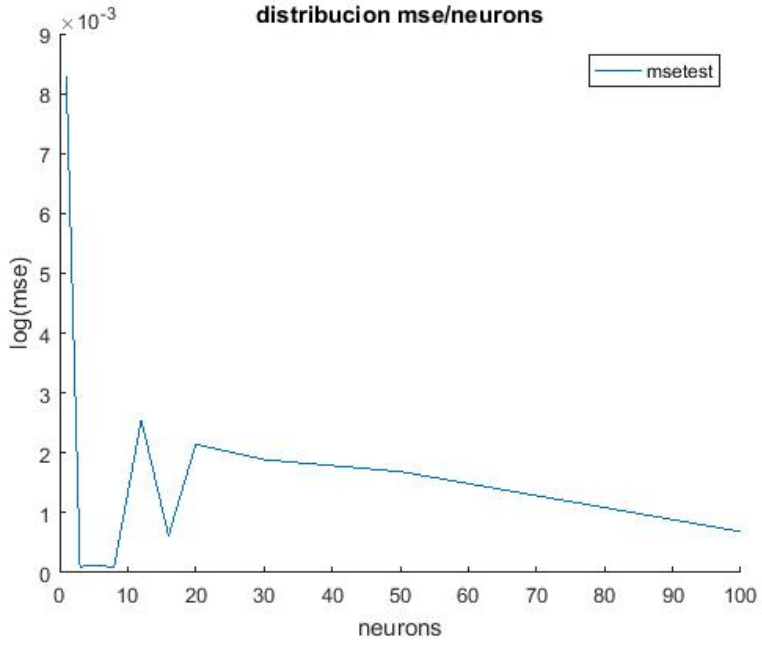
At this point, we noticed that we need a training strategy to calibrate the weights . It exists several training algorithms (Levenberg-Marquardt, Bayesian Regularization, Scale Conjugate Gradient). We will use the uploaded script (Levenberg-Marquardt) of our material with the following parametres.

-input training set size =40.

-iterations =100.

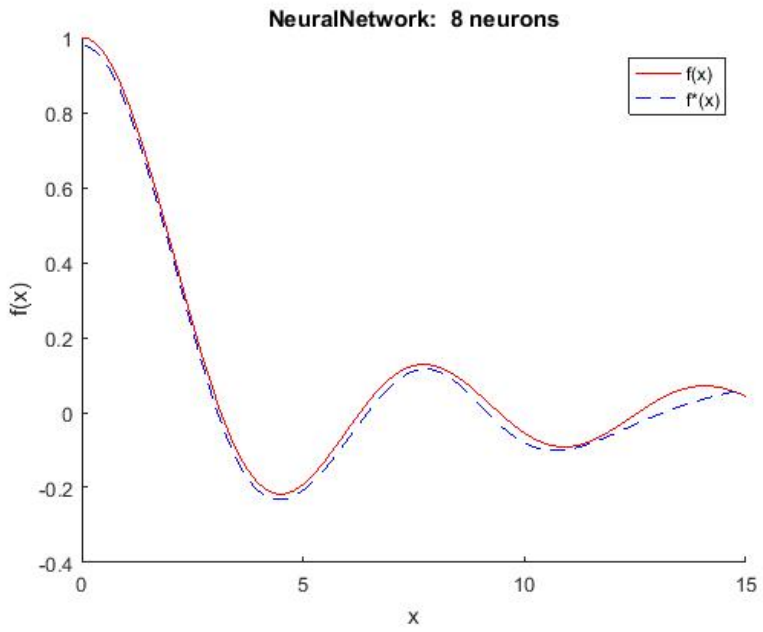
-input test set size = 400.

Number of hidden network	Training set mse	Test set mse
1	0,00814	0,00827
3	0,00012	0,00010
6	0,00012	0,00011
8	0,00010	9,05183e-05
12	0,00252	0,00255
16	0,00060	0,00061
20	0,00209	0,00214
30	0,00184	0,00188
50	0,00166	0,00168
100	0,00074	0,00068



How we can appreciate, the best result was obtained using 8 neurons. In the following image it is representring the structure of the best neural network and several plots that show the datas about this work.

Rubén Romero Hernández **rromero@mion.elka.pw.edu.pl**



It is really astonishing how neural networks work. In my opinon it is a really creative desing, and it has myriads application in the today´s world. We were able to estimate very good outcomes with a small error even though one of our inputs was a random value. The most important and more difficult of the neural network is the training. During my test, training my neural network using Levenberg-Marquardt algorithm, I notice that there is not a big different between the mse using among 4 and 100 neurons, however there is a considerable different in the time spent during the work due to the complexity of the calculations that our neural network requires to do. So that, it is important to minimize the mse using the minor number of hidden neurons possible. The order of efficiency of this neural network is :

m= number of hidden neurons

n= number of input

$O(n,m)= nm^2$

Time spending in training => $T(nºtrain, n, m) = nºtrain*n*m^2$

Also, I tried to see the overfitting with different datas, but even with 1000 neruon was impossible to get a clearly overfitting.

References:

I used matlab and its different tool for neural network (train_net…) to develop this project. Moreover, it was neccesary to use the script provided in moodle to understant how using these tools and write other script to get the required outcomes.

This pdf will be send with the pertinet script and plots.