Practice 1
Neural Networks

Mariajose Franco Orozco
Mathematical Engineering
EAFIT University
Medellín, Colombia
mfrancoo@eafit.edu.co

I. Introduction

Artificial Intelligence (AI) is a field of computer science capable of mimicking human reasoning (IBM, 2022). AI involves the development of intelligent machines that can perform tasks that typically require human intelligence, such as learning, reasoning, problem-solving, and many others (Ramesh *et al.*, 2004).

There are different approaches to AI, including Neural Networks, which are algorithms to imitate the human brain through something called 'artificial neurons' which consist of multiple nodes or neurons that are interconnected to other multiple layers of a similar structure and which are capable of learning from the data in order to transmit information. These neurons receive some inputs from other neurons or external sources, process them with an activation function, and produce an output that is passed on to other neurons or to give the estimated outputs (Quintero, 2020). Neural Networks make a learning process to find the best weights for the network that improves their performance, which allows them to make more accurate predictions.

The perceptron, as it was mentioned in class, is the smallest unit of a Neural Network and was the first model of a neuron proposed by Frank Rosenblatt. A multilayer perceptron, also known as MLP, is a type of neural network that consists of sets of layers that contain sets of neurons that are fully interconnected and have the purpose of processing data to transmit a possible output. The main feature of an MLP is that every layer can have a different activation function, and finally, the idea is to minimize the error between the desired output and the estimated output by finding the best weights for every layer with a process called backpropagation. In the following section, the concepts used for implementing an MLP algorithm are described.

II. Concepts and Definitions

The index i represents the input layer, the index j the hidden layers, and the index k the output layer.

A. Inputs and Outputs

The inputs for the neurons i are represented as $x_i(n)$ and the outputs of the neurons k are represented as $y_k(n)$ and are defined as

$$x_i, y_k \in [-1, 1] \tag{1}$$

B. Weights

The weights are the values that are assigned to a connection between 2 neurons. The weight that connects the neuron i with neuron j is

$$w_{ji}(n) \tag{2}$$

C. Local Field

Is the linear combination of the inputs and weights. The local field of the neuron k is defined as

$$v_k(n) = \sum x_k w_{kj} \tag{3}$$

D. Activation Function

The processing of the local field can be done by the activation function. Several functions can be used, but the most used functions are, the linear function, the sigmoid function, the ReLU function, and the hyperbolic tangent function. It is defined as

$$\phi(v) \tag{4}$$

The activation function applied to the local field produces the output of the respective neuron. The output of a neuron i is the input of a neuron j, the output of a neuron j is the input of a neuron k, and the output of the neuron k is the estimated output of the MLP, and is defined as

$$y_k = \phi_k(v_k) \tag{5}$$

E. Output Error

Measures the difference between the real or desired output and the estimated output. It is defined as

$$e_k(n) = y_k(n) - \hat{y_k}(n) \tag{6}$$

It is only possible to measure the error in the output layer because it is where the estimated output is obtained.

F. Instant Energy Error

It is defined as

$$\varepsilon(p) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(p) \tag{7}$$

G. Average energy

It is defined as

$$\varepsilon_{av} = \frac{1}{N}\sum_{p=1}^{n} \varepsilon(p) \tag{8}$$

H. Local Gradient

The local gradient is calculated differently for the output layer and the hidden layers because in the output layer is possible to calculate the error of the approximation, but in the other layers it can not be done.

For the output layer, the local gradient is calculated as follows

$$\delta_k = e_k \phi_k^{'}(v_k) \tag{9}$$

And, for the hidden layers, the local gradient is

$$\delta_j = \phi_j^{'}(v_j) \sum \delta_k w_{kj} \tag{10}$$

I. Weights Update

The MLP tends to find the best weight that minimizes the error to give an accurate output. The weights are initialized randomly but they are updated according to the learning process that MLP does. The update of the weights that connect the layer j with the layer k is calculated as follows

$$\Delta w_{kj}(n) = -\delta_k Y_j \tag{11}$$

And for finding the new weights, a learning rate (η) is defined.

$$w = w + \eta \Delta w \tag{12}$$

III. Methodology

The data used in this practice was obtained from the dataset posted by the teacher on the 'teams' platform. This dataset contains 3.001 rows of data and 3 columns. For this practice, 300 rows were selected, the first and the second columns are the inputs of the MLP, and the last column is the output. According to this, this MLP will contain 2 inputs and 1 output as it is presented in Figure 1.

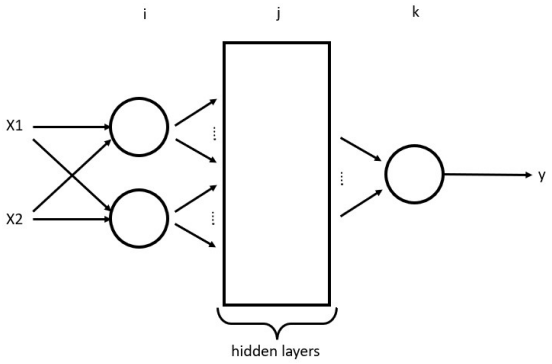

Fig. 1. Multilayer Perceptron

The maximum number of iterations considered is 50, and the activation function implemented is the sigmoid function. Some parameters that are being considered are the learning rate, the number of hidden layers, and the number of neurons on the hidden layers. For the learning rate, 3 possible

options were tested: $0.2, 0.5$, and 0.9. The number of hidden layers could be from 1 to 3, and the number of neurons on the hidden layers could be from 1 to 5. All the possible combinations are considered, obtaining 45 different models. In the following sections, the experimentation and the results of the models tested are presented.

IV. Experimentation

The dataset used contains 300 data as it was mentioned before. This data is partitioned into 3 subsets: 60% for training, 20% for testing, and 20% for validation.

In Figure 2, we can see the performance of the different models considered.

index	Learning Rate	Architecture	Average Energy
0	0.2	2,1,1	0.0370257792
1	0.2	2,2,1	0.0366892475
2	0.2	2,3,1	0.039827522
3	0.2	2,4,1	0.0388459997
4	0.2	2,5,1	0.0321847205
5	0.2	2,1,1,1	0.040243343
6	0.2	2,2,2,1	0.0401391306
7	0.2	2,3,3,1	0.0402593035
8	0.2	2,4,4,1	0.0402724089
9	0.2	2,5,5,1	0.0794187179
10	0.2	2,1,1,1,1	0.040286124
11	0.2	2,2,2,2,1	0.0403103101
12	0.2	2,3,3,3,1	0.0403067643
13	0.2	2,4,4,4,1	0.0403103188
14	0.2	2,5,5,5,1	0.0403350889
15	0.5	2,1,1	0.0112409326
16	0.5	2,2,1	0.0115813719
17	0.5	2,3,1	0.0111573153
18	0.5	2,4,1	0.0112620113
19	0.5	2,5,1	0.011586199
20	0.5	2,1,1,1	0.0402796466
21	0.5	2,2,2,1	0.0374939407
22	0.5	2,3,3,1	0.0224655097
23	0.5	2,4,4,1	0.0970854823
24	0.5	2,5,5,1	0.0390516183
25	0.5	2,1,1,1,1	0.0403019361
26	0.5	2,2,2,2,1	0.0401886742
27	0.5	2,3,3,3,1	0.0401663244
28	0.5	2,4,4,4,1	0.0365293297
29	0.5	2,5,5,5,1	0.0398629302
30	0.9	2,1,1	0.0108193739
31	0.9	2,2,1	0.0108932847
32	0.9	2,3,1	0.0110878265
33	0.9	2,4,1	0.0110313684
34	0.9	2,5,1	0.0110894567
35	0.9	2,1,1,1	0.0112062873
36	0.9	2,2,2,1	0.0180583724
37	0.9	2,3,3,1	0.0111602089
38	0.9	2,4,4,1	0.0186529036
39	0.9	2,5,5,1	0.0118720836
40	0.9	2,1,1,1,1	0.040288153
41	0.9	2,2,2,2,1	0.0433409039
42	0.9	2,3,3,3,1	0.0397356378
43	0.9	2,4,4,4,1	0.0409758064
44	0.9	2,5,5,5,1	0.0535137703

Fig. 2. Results of training 45 models

The following figures present the performance of 3 randomly selected models in order to show the behavior of the MLP algorithm.

A. Learning rate 0.2 and 3 hidden layers

In the following figures, the local gradient and the average error for this model are presented.

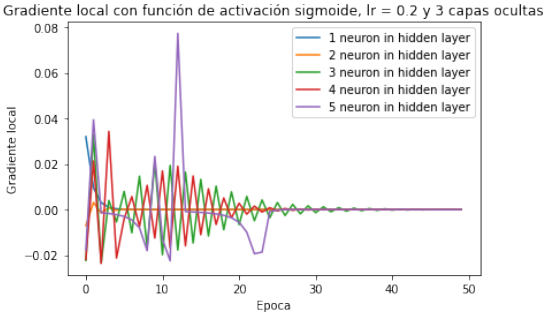

Fig. 3. Local Gradient with a learning rate 0.2 and 3 hidden layers

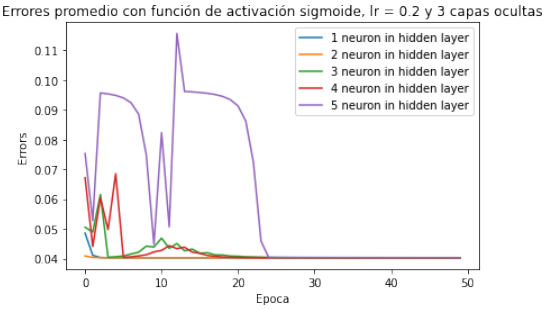

Fig. 4. Average error with a learning rate 0.2 and 3 hidden layers

B. Learning rate 0.5 and 1 hidden layer

In the following figures, the local gradient and the average error for this model are presented.

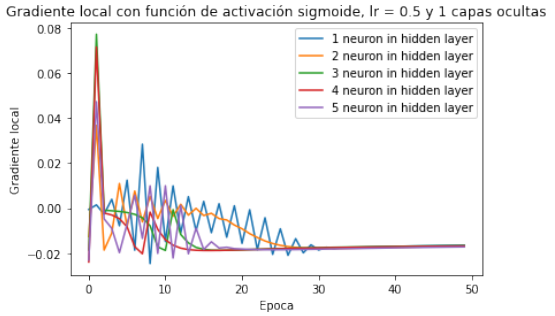

Fig. 5. Local Gradient with a learning rate 0.5 and 1 hidden layer

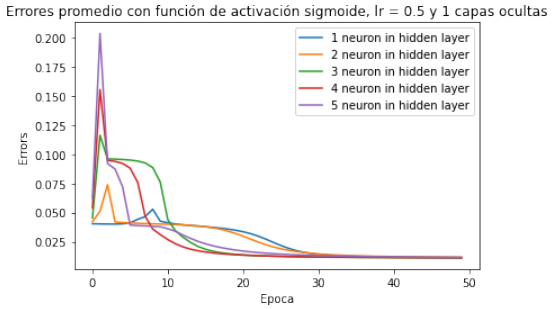

Fig. 6. Average error with a learning rate 0.5 and 1 hidden layer

C. *Learning rate 0.9 and 2 hidden layers*

In the following figures, the local gradient and the average error for this model are presented.

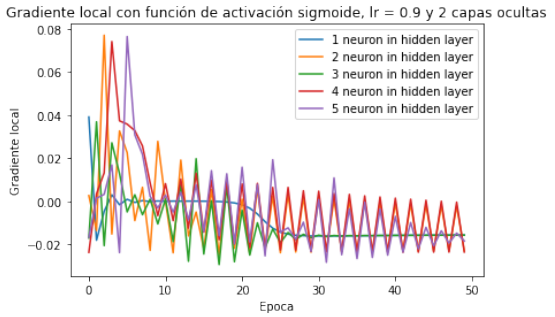

Fig. 7. Local Gradient with a learning rate 0.9 and 2 hidden layers

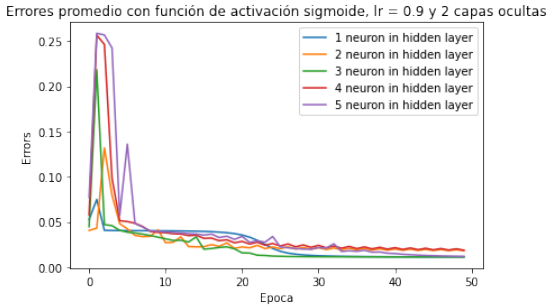

Fig. 8. Average errors with a learning rate 0.9 and 2 hidden layers

V. Results

After obtaining the different results with the MLP algorithm, the next step was to determine which is the best model, the worst model, and the average model.

Here, will be presented the 3 best, worst, and average models.

Arquitecture	Learning Rate	Average Error
[2, 1, 1]	0.9	0.010819
[2, 2, 1]	0.9	0.010893
[2, 4, 1]	0.9	0.011031

TABLE I
BEST 3 MODELS

Arquitecture	Learning Rate	Average Error
[2, 4, 4, 1]	0.5	0.097085
[2, 5, 5, 1]	0.2	0.079418
[2, 5, 5, 5, 1]	0.9	0.053513

TABLE II
WORST 3 MODELS

Arquitecture	Learning Rate	Average Error
[2, 5, 1]	0.2	0.032184
[2, 4, 4, 4, 1]	0.5	0.036529
[2, 3, 3, 1]	0.5	0.022465

TABLE III
AVERAGE MODELS

The best model is the one with a learning rate of 0.9, 3 layers, and 1 neuron in the hidden layer. The worst model was the model with a learning rate of 0.5, 4 layers, and 4 neurons in the hidden layers. Finally, the average model has a learning rate of 0.2, 3 layers, and 5 neurons in the hidden layer.

VI. Discussion

It can be concluded that the results obtained are not the expected outputs because of the poor prediction of the algorithm, but in this practice, bias was not considered, which means that the performance of the MLP can be improved. In the table of the 3 best models is possible to see that the 3 of them were with a learning rate of 0.9 and with 3 layers. Personally, I would think that with more hidden layers and more neurons in it, the MLP predictions will be better, but this results showed me that with less layers, the prediction was more precise, this is something I would like to learn more about in order to analyze if the results presented here are correctly. In spite of the results, this practice allowed me to understand the behavior of an artificial neural network and learn about how it is done. As I was implementing it, I found myself with several errors that allowed me to review the concepts again and correct them for a properly implementation.

References

IBM. 2022. *Artificial Intelligence*.

Quintero, O Lucia. 2020. Machine Intelligence for Human Decision Making.

Ramesh, A. N., Kambhampati, C., Monson, J. R.T., & Drew, P. J. 2004. Artificial intelligence in medicine. *Annals of the Royal College of Surgeons of England*, **86**(5), 334–338.