MultiLayer Perceptron Report

Abelino Sepúlveda Estrada
Universidad EAFIT
Medellin, Colombia
asepulvede@eafit.edu.co

Olga Lucía Quintero Montoya
Universidad EAFIT
Medellin, Colombia
oquinte1@eafit.edu.co

***Abstract*—This paper investigates the impact of different Multilayer Perceptron (MLP) architectures on the learning and training processes, employing 45 models with varying parameters, including hidden layers, neurons per layer, and learning rates. The results indicate that, in general, architectures with fewer neurons per layer outperform others, though this outcome may vary with different datasets. Therefore, thorough experimentation is crucial to determine the most suitable MLP architecture for specific data.**

***Index Terms*—Neural Network, MultiLayer Perceptron, Model Construction**

I. Introduction

Artificial Neural Networks (ANNs) have become a critical artificial intelligence paradigm, providing a flexible framework for resolving challenging issues in various domains. Multilayer Perceptrons (MLPs) are a fundamental and frequently used class of neural networks among the various architectures. Layered structures, such as those found in MLPs, allow for the processing of information by interconnected nodes (also known as neurons) arranged in various layers. Each layer helps to extract hierarchical features, which enables MLPs to recognize complex patterns in data. In order to fully understand Multilayer Perceptrons, this report aims to explain their underlying principles, training procedures, and wide range of applications.

The ability of Multilayer Perceptrons to approximate complex functions by combining simpler, non-linear transformations makes them a prime example of feedforward neural networks. This makes it possible for them to excel at more complex tasks like image recognition, natural language processing, and tasks ranging from regression to classification. Understanding MLPs' inner workings is essential for grasping the larger picture of neural networks and maximizing their potential in real-world applications across various industries. We will examine the architecture, activation functions, and training algorithms of MLPs as we delve into their complexity, laying a solid foundation for future research in neural networks and AI.

The structure of the paper is as follows: methodology, where the data is discussed and defined; the architecture, the optimization method, and the experimentation are defined; the results, where it is shown what was obtained from the experimentation; and finally, the conclusions.

II. Methodology

A. Data and Problem

The data selected for this work consists of two input features and an output with 3001 observations of a dynamic phenomenon. The nature and the actual context of the data are currently unknown. However, it is observed that all the variables are numerical and that they are on different scales.

As all features have variable scales, the data are taken to the hyperplane $[0, 1]$, i.e., normalized. This is to avoid numerical problems, improve convergence, and make the training less dependent on the initialization of the data.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

The equation (1) refers to the normalized data by means of min-max normalization. Where X_{norm} refers to the normalized data, X the original data and X_{min}, X_{max}, the minimum and maximum of the features respectively.

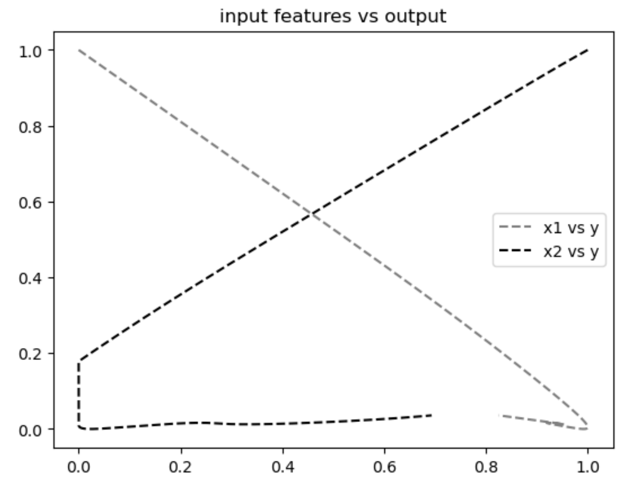

Fig. 1: Data visualization: output vs input features

Subsequently, the data are separated into training, testing, and validation sets in a proportion of $60\%, 20\%, and 20\%$ of the original data, respectively. This avoids over-fitting and allows us to assess the model's generalization ability to new data. Finally, this partitioning was done based on the uniform distribution, i.e., the probability of selecting each data set was the same to guarantee a higher entropy and that the sets are representative of the general distribution of the original data.

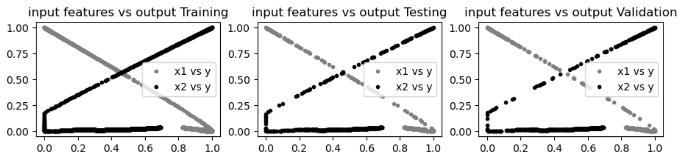

Fig. 2: Set partitioning of data

Figure (1) shows the original data in the hyperplane $[0, 1]$ and figure 2 shows that the sampling of the different sets has the same behavior as in figure 1, so we can say that they are a representative sample of the data.

Finally, for academic purposes, the selected problem for this work is a curve-fitting problem.

B. MultiLayer Perceptron Architectures

A multi-layer perceptron (MLP) is an artificial neural network composed of multiple layers of processing units called neurons. The structure is divided into an input layer, one or more hidden layers, and an output layer. This network is a fully connected architecture that receives input data and transforms them into a set of possible outputs. The architecture can be seen in figure 3

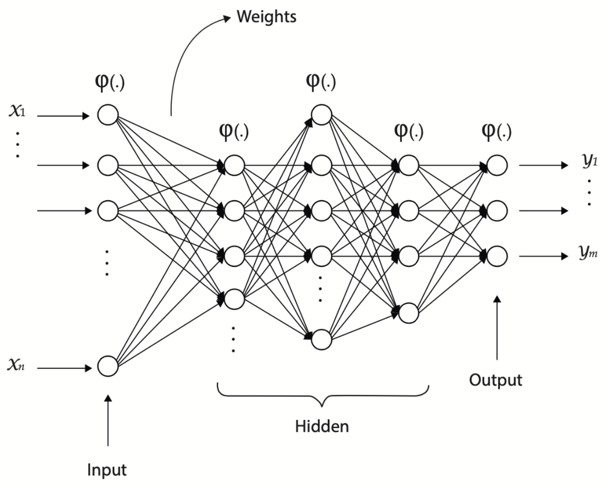

Fig. 3: Multilayer perceptron architecture

Where:

- x_1, ..., x_n are the inputs of the model.
- y_1, ..., y_m the outputs.
- $\phi(.)$ the activation functions of each layer

Also, we denote w_i as the weigths of the layer i.

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i\right) \tag{2}$$

The model's output is a product of forward propagation, where the input data passes through the network's layers, ultimately determined by the neurons in the last layer. This process, often seen in an MLP (Multilayer Perceptron), involves the propagation of input information through the model as explained in Equation (2). Notably, until this point, no learning has occurred; the input is transmitted through the network to the output layer.

C. Backpropagation and optimization method

Backpropagation is a learning technique that allows the multilayer perceptron to iteratively adjust the weights to bring the cost function (energy error) to zero. This error is defined in the following equation.

$$\mathscr{E}_{av} = \frac{1}{N}\sum_{n=1}^{N}\mathscr{E}(n) \tag{3}$$

Where

$$\mathscr{E}(n) = \frac{1}{N}\sum_{k\in L} e_k^2(n) \tag{4}$$

is the instantaneous energy error of the datapoint n and where

$$e_k(n) = y_k(n) - y_k(n) \tag{5}$$

is the difference between the output $y_k(n)$ and the desired state $d_k(n)$.

The main objective is to adjust the weights for each iteration that minimize the average estimation error in Equation (4). For this, we make use of the gradient descent optimization method.

$$\Delta w_{ji} = -\eta \delta_j y_i \tag{6}$$

where η is the learning rate, δ_j the local gradient of the layer j and y_i is the output of the layer i.

D. Experimentation

The experiment used a fully connected perceptron with $L = 1, 2, 3$ hidden layers, $l = 1, 2, 3, 4, 5$ neurons in each layer, and $\eta = 0.2, 0.5, 0.9$. Where $N = 3001$, $m = k = 1$. Additionally, the model is trained for 50 epochs, and ϵ is calculated with the validation set to select the models. Where

$$\epsilon = p(h\Delta c) \tag{7}$$

III. Results

Table I shows the results obtained and the architectures of each of the models.

L	l	# Parameters	η	T error	t error	ϵ
1	1	7	0.2	0.403	0.406	0.92
			0.5	0.409	0.391	0.903
			0.9	0.403	0.409	0.923
	2	10	0.2	0.407	0.406	0.9
			0.5	0.399	0.415	0.918
			0.9	0.406	0.411	0.915
	3	13	0.2	0.406	0.402	0.928
			0.5	0.402	0.406	0.911
			0.9	0.405	0.407	0.921
	4	16	0.2	0.405	0.402	0.898
			0.5	0.406	0.394	0.926
			0.9	0.406	0.405	0.908
	5	19	0.2	0.406	0.407	0.896
			0.5	0.408	0.389	0.918
			0.9	0.406	0.403	0.91
2	1	8	0.2	0.404	0.407	0.925
			0.5	0.404	0.407	0.91
			0.9	0.406	0.398	0.92
	2	14	0.2	0.406	0.398	0.918
			0.5	0.409	0.397	0.908
			0.9	0.406	0.413	0.905
	3	22	0.2	0.403	0.413	0.915
			0.5	0.401	0.410	0.921
			0.9	0.403	0.410	0.911
	4	32	0.2	0.407	0.408	0.906
			0.5	0.407	0.397	0.91
			0.9	0.407	0.399	0.915
	5	44	0.2	0.401	0.413	0.928
			0.5	0.405	0.406	0.901
			0.9	0.405	0.414	0.91
3	1	9	0.2	0.404	0.407	0.918
			0.5	0.403	0.417	0.908
			0.9	0.404	0.407	0.926
	2	18	0.2	0.407	0.403	0.91
			0.5	0.406	0.406	0.908
			0.9	0.401	0.405	0.935
	3	31	0.2	0.404	0.413	0.913
			0.5	0.406	0.403	0.915
			0.9	0.403	0.420	0.903
	4	48	0.2	0.409	0.401	0.91
			0.5	0.402	0.418	0.906
			0.9	0.406	0.398	0.92
	5	69	0.2	0.408	0.401	0.908
			0.5	0.403	0.410	0.925
			0.9	0.408	0.399	0.915

TABLE I: Results

Where T error and t error are the training and testing errors, respectively, we can also see that the best, worst, and average models are highlighted on a grey scale. Where the lightest grey represents the best and the darkest the worst. Finally, in figure 7, we show the e plot for each model where the three models obtained above are highlighted.

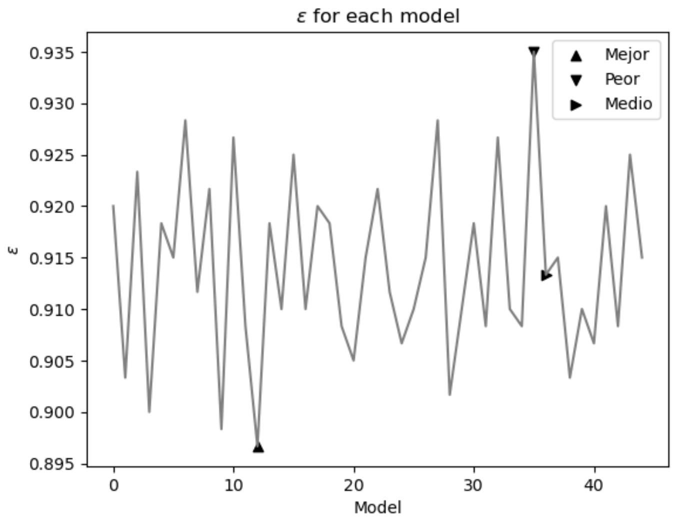

Fig. 4: $epsilon$ for each model

IV. Discussion and Conclusion

After experimentation, we can draw multiple conclusions: We know that fewer neurons per layer make it a shallow network with fewer hyperparameters to fit, which makes the model less complex following the principle of parsimony. Our results show that the models obtained (the best, the worst, and the average) follow this theory.

We observe that the errors tend to be very high; this may be due to the lack of bias, as the models were implemented without it, as the models may need help in modeling complex relationships and capturing patterns in the data.

These errors may also have been due to a need for more hyperparameter tuning for the models, such as the parameters for the activation functions of each layer.