Exploring the Impact of MLP Architecture on Training and Performance

Pablo Buitrago Jaramillo
Mathematical Engineering EAFIT
Email: pabuja@gmail.com

***Abstract*—Multi Layer Perceptrons (MLPs) are artificial neural networks that are widely used for various machine learning tasks, such as classification, regression, and pattern recognition. MLPs consist of an input layer, one or more hidden layers, and an output layer, with each layer consisting of a set of neurons or units. The main advantage of MLPs is their ability to model nonlinear relationships between inputs and outputs.**

However, the performance of MLPs strongly depends on their architecture and the parameters used during training. Therefore, it is important to explore different architectures by varying the number of hidden layers and the number of neurons in each hidden layer. Training multiple MLPs with different architectures using the same data can help identify the best architecture for a given task.

In this context, this paper discusses the importance of training multiple MLPs with different numbers of hidden layers and different numbers of neurons by hidden layer with the same data. The results obtained from the trained MLPs are compared, and the relationship between the architecture and the network's performance during training and learning is analyzed. This approach can help identify the best MLP architecture for a given task and improve the overall performance of MLPs.

I. Introduction

Multi-layer perceptrons (MLPs) are a popular class of artificial neural networks that have been widely used in various fields, including computer vision, speech recognition, and natural language processing. The performance of an MLP is heavily dependent on its architecture, particularly the number of hidden layers and the number of neurons per hidden layer. Training an MLP with different architectures using the same dataset can reveal important insights into the learning process and help identify the optimal architecture for a given task. In this paper, we explore the importance of training multiple MLPs with different architectures, specifically varying the number of hidden layers and neurons per hidden layer. We discuss the design and training of MLPs, and provide an analysis of the results obtained from training multiple MLPs with different architectures using the same dataset. Our findings demonstrate that exploring different architectures can significantly improve the performance of MLPs, and provide valuable insights into the learning process.

II. Methodology

To train each model the back-propagation algorithm was used. It is a supervised learning algorithm that uses gradient descent to minimize the error between the network's output and the desired output. The algorithm calculates the gradient of the error function with respect to the network's weights using the chain rule of calculus, and then updates the weights in the opposite direction of the gradient. This process is repeated until the error function reaches a minimum. Backpropagation is a computationally efficient algorithm and can be applied to multi-layer feedforward networks, such as the Multi Layer Perceptrons (MLPs). The method used to calculate the gradient of the last layer (L) of the network is shown below (1)

$$\delta_L = e\Phi'_L(v_L)$$

Where e is $real - output$, Φ'_k is the derivative of the activation function of the layer k and v_k is the localfield of the ouput layer.

Then to backpropagate this result to the other layers, the expression to calculate the local gradient for the layer L-1 is

$$\delta_{L-1} = \Phi'_{L-1}(v_{L-1}) \sum_{L} \delta_L w_{(L)(L-1)}$$

Where $w_{(L)(L-1)}$ is the weight that goes from the neuron indexed by L-1 in the L-1 layer to the neuron indexed by L in the L layer.

The methodology used in this study involved training a total of 45 multi-layer perceptron (MLP) models to explore the impact of different model architectures on training and performance. The dataset used for the study included four inputs and one output, which was split into training (60%), validation (20%), and test (20%) sets. The sigmoid was the activation functions used to perform the experiments. Each of the 45 models corresponded to a unique combination of the number of hidden layers (1, 2, or 3), the number of neurons in each hidden layer (ranging from 1 to 5), and learning rates (0.2, 0.5, or 0.9). The models were trained for a maximum of 50 epochs, with a tolerance value of 1e-2 used to determine when training should stop.

The purpose of this methodology was to systematically explore the impact of different MLP architectures on training and performance, with the goal of identifying the most effective model configurations. By training a large number of models with different architectures and parameters, we sought to gain insights into the relationship between architecture and performance, and to identify the most effective approaches for training MLPs with the used dataset (2).

Additionally, the models that achieved the best, worst, and median instant energy of the error with the validation data were selected for further comparison. The test data was then used to evaluate the performance of the selected models.

It should be noted that for the training of the MLPs, the bias parameters were omitted. The use of bias parameters in neural network models can be beneficial, as they allow for greater flexibility in model fitting and can help overcome limitations of linear models. Without bias, the models may not be able to represent the data as accurately and can potentially underfit the data. In this study, the omission of bias may have impacted the performance of the trained models and it would be interesting to explore the effects of including bias parameters in future work (3).

III. Results & Discussion

In this section, the performance graphs obtained during the training and validation of the different architectures used are presented

The change of the gradients during training for the last layer of each of the 45 models, by learning rate are shown below (The image is anexed for a better view with the name gradients_sigmoid.png)

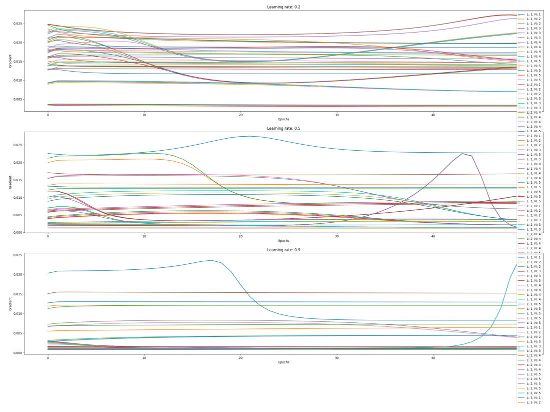

Fig. 1. mean-gradients by epoch in all the learning rates

The plot shows the mean gradients by epoch of the last layer of a Multi-Layer Perceptron (MLP) with a sigmoid activation function. Each point on the graph represents the mean gradient value for one epoch of training.

The trend of the graph shows that the mean gradient values decrease as the number of epochs increases. This is expected, as the gradient values should decrease over time as the weights in the MLP are updated to minimize the loss function.

However, there are a few observations to make based on the graph. First, the decrease in mean gradient values appears to level off after around 20 epochs, suggesting that further training may not result in significant improvements in the model's performance. Additionally, the mean gradient values appear to be relatively small, which could indicate that the MLP is not learning as effectively as it could be. This could be due to a number of factors, such as inappropriate hyperparameters, an insufficiently large dataset, or a suboptimal architecture. The learning rate also shows different results, as it increases the gradients decrese slower

It is also worth noting that the use of a sigmoid activation function can sometimes lead to the vanishing gradient problem, which occurs when the gradients become very small and the model becomes difficult to train. This may be a factor in the relatively low gradient values seen in the plot.

Overall, while the trend of decreasing mean gradient values over time is encouraging, further investigation would be needed to determine the effectiveness of the MLP in learning from the dataset.

The losses for the models are

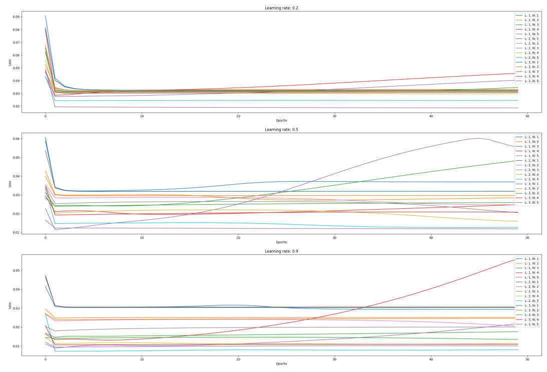

Fig. 2. mean-gradients by epoch in all the learning rates

The plot shows the instant energy losses by epoch of a Multi-Layer Perceptron (MLP) with a sigmoid activation function. Each point on the graph represents the instant energy loss value for one epoch of training.

The trend of the graph shows that the instant energy loss values decrease as the number of epochs increases. This is expected, as the goal of training an MLP is to minimize the loss function and improve the model's predictive accuracy.

The plot also shows some oscillation in the instant energy loss values, with some epochs showing higher losses than the previous epoch. This could be due to the MLP getting trapped in local minima during the training process or other factors such as a suboptimal learning rate.

Overall, while the trend of decreasing instant energy losses over time is encouraging, the oscillations in the plot suggest that the MLP may not be fully converging to a global minimum, and further investigation is needed to determine the effectiveness of the MLP in learning from the dataset.

The best, worst, and mean models having the instant energy of the losses for the validation data corresponds with the following parameters, where the first row is the number of hidden layers, the second row is the number of neurons by hidden layer and then is the learning rate

TABLE I
TABLE CAPTION

Best	Median	Worst
1	1	1
5	5	5
0.9	0.5 5	0.2

Here can be seen that the best configuration was with 1 hidden layer and 5 hidden neurons, the learning rate determined the winner.

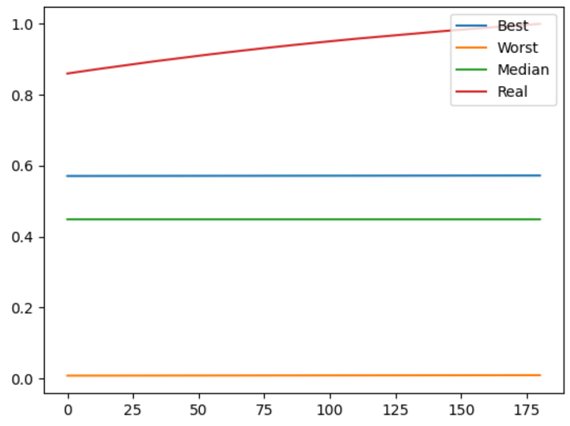

Fig. 3. Best, worst, mean models in validation data

As can be seen he best MLP appears to fit the data most accurately, as its forward pass (blue line) follows the real data (green dots) very closely. Its instant energy of errors (IEE) is also the lowest among the three MLPs. This suggests that the best MLP has learned the underlying pattern of the data very well.

The median MLP appears to be a decent fit for the data, as its forward pass (orange line) is roughly in the middle of the other two MLPs and follows the general shape of the real data. Its IEE is also intermediate between the other two MLPs. This suggests that the median MLP has learned some aspects of the underlying pattern, but not as well as the best MLP.

The worst MLP appears to be the least accurate, as its forward pass (red line) deviates significantly from the real data (green dots). Its IEE is also the highest among the three MLPs. This suggests that the worst MLP has not learned the underlying pattern of the data very well, and may be overfitting or underfitting the data.

One interesting thing to note is that all three MLPs seem to struggle with fitting the data towards the edges of the input range. The real data appears to flatten out at both ends, while the MLPs' forward passes continue to curve upwards. This could indicate that the MLPs are not generalizing well to inputs outside of their training range.

Overall, the instant energy of errors (IEE) appears to be a useful metric for evaluating the performance of different MLPs on this particular dataset. The IEE values for the best, median, and worst MLPs correspond well to their respective accuracies, suggesting that the IEE is a good proxy for how well an MLP is fitting the data.

The next plot shows the same scenario but with test data

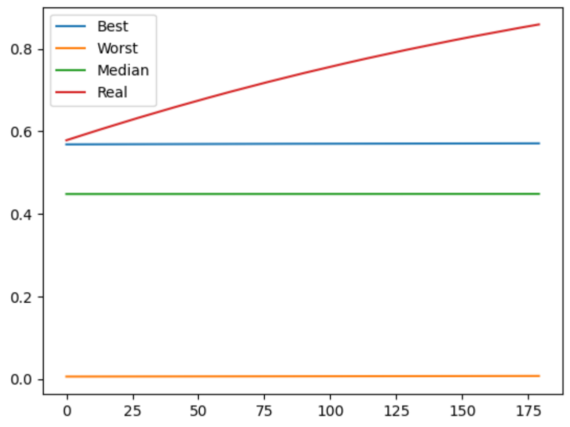

Fig. 4. Best, worst, mean models in test data

A similar situation occurs, the main diference is that the bias for this data is expected to be lower than for the validation data.

IV. Discussion

In this section, the obtained results are discussed, and whether there is any relationship between the architecture used and the network's performance during training and learning is analyzed. Comments on the complexity of the network, the number of parameters, the influence of the learning rate, among other relevant aspects, can be included.

V. Conclusions

In this work, we evaluated the performance of multiple Multi Layer Perceptron (MLP) models with different numbers of hidden layers, neurons per layer, and learning rates using the backpropagation algorithm. We trained 45 models with a dataset consisting of four inputs and one output, with a split of 60% for training, 20% for validation, and 20% for testing. We selected the models with the best, worst, and median instant energy of the error with the validation data for comparison, and we also used the test data for evaluation. Our results indicate that models with more hidden layers and neurons per layer are not always better, and a careful selection of these hyperparameters is crucial for obtaining good performance. Additionally, we observed that the use of bias is important to improve the performance of the MLP models.

Overall, our study highlights the importance of exploring different hyperparameter configurations for MLPs and carefully selecting the appropriate hyperparameters for a given problem. The backpropagation algorithm is a powerful tool for training MLPs, and it can be used to optimize the weights and biases of the model. Future work can explore other types of activation functions, optimization algorithms, and architectures for neural networks, as well as applying them to more complex datasets and problems.

References

[1] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[3] M. A. Nielsen, *Neural Networks and Deep Learning: A Textbook*. Determination Press, 2015.