Multilayer Perceptron: Training analysis

Ana Sofia Gutierrez Tejada
Universidad EAFIT
Medellin, Colombia
asgutierrt@eafit.edu.co

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

***Abstract*—This project is a learning exercise about training of a multilayer perceptron (MLP), a type of artificial neural network. It also encompasses the development of criteria to confidently make deductions about the training process and characteristics. The methodology called gradient descent looks to minimize errors in the modeled output in an iterative way. Additionally, we correlate the results to the learning problem that motivates the use of a MLP model. The work is based on non-standard data provided as part of our course, giving us a chance to put our theoretical knowledge into practice.**

***Index Terms*—Neural Network, Multilayer Perceptron, Iterative trainning**

I. Introduction

This project explores the intricacies of implementing the multilayer perceptron (MLP) model. A widely employed technique based on making data-driven inferences but requiring meticulous refinement. Pattern recognition models, such as MLPs, have gained significant prominence and widespread application across various domains. Our proposition centers on the training of an MLP using the gradient descent technique through the backpropagation algorithm.

Nonetheless, the widespread adoption of this model has often resulted in a loss of precision in its implementation, which ultimately holds significance. This project represents a subsequent phase following a simplified preliminary implementation. The primary objective here is to comprehend the underlying nature of the model and its training mechanism. We conceptualize the model as the means to achieve a learning goal and the training technique as the method to accomplish a training objective. By doing so, we aim to gain a deeper insight into the intricate interplay between the model's architecture and the training techniques employed to optimize its performance.

The next sections of this academic work will delve into the setup of the Multilayer Perceptron (MLP) and the Backpropagation technique. Both have been implemented using the dataset provided in the course. The latter sections will then examine the outcomes derived from our experimentation with the data and initiate a discussion based on these results. Finally, the closing section states the conclusions from the proyect.

II. Methodology

A. MLP Model statement

In our study, we employ fully connected Multilayer Perceptron (MLP) models, varying the architecture from 1 to 3 hidden layers and 1 to 5 neurons per hidden layer. Given that the input data has 2 variables, the input layer is proposed as a 2-neurons layer. Similarly, as the output consist of single-dimensional data, the output layer is configured with only one neuron. To provide further clarity, the mathematical expression for the neurons on each layer j can be outlined as follows:

$$Y_j = \phi(\nu(X_j))$$

$$with\ \nu(W, X_j) = WX_j\ and\ \phi(X) = \frac{1}{1+e^{-X}}$$

The position of each neuron within the network is determined by its input. Neurons situated in the input layer receive two variables of a data-point X_p as their input. Neurons in the other layers receive an input with as many variables as there are neurons in the preceding layer. Within this framework, we defined the functions for the local field v and activation phi for each layer, and it's worth noting that these play a significant role in shaping the outcomes of the MLP.

B. Training the model

In order to apply the model effectively, the weights denoted as W that define the local field need to be determined. We pose an optimization problem based on the effectiveness of the output neurons in aligning their output Y_j with the desired state Y_{dj}, given a specific stimulus on the input layer X_p. As a cost function, we define the instantaneous error energy for an input X_p follows:

$$\mathscr{E} = \frac{\sum_j (Y_{dj} - Y_j)^2}{2}$$

The update rule for the weights W_j in each layer is determined by the gradient of the error with respect to each weight, as follows:

$$W_j* = W_j + \eta \frac{\partial \mathscr{E}}{\partial W_j}$$

The expression for $\frac{\partial \mathscr{E}}{\partial W_j} = \delta_j X_j$, where X_j represents the input to that specific layer. This equation highlights the role of the gradient δ_j in adjusting the weights to minimize the error, with X_j serving as the input signal guiding this adjustment process.

It is essential to note that error measurement values are only available at the output layer of the model. The backpropagation algorithm addresses this challenge by propagating the error energy computed at the output layer backward through the hidden layers. This involves applying the chain rule in the differentiation of the functions defining the local fields of the internal neurons. Ultimately, the expression for the gradient in each neuron not located in the output layer is determined by:

$$\delta_j = \begin{cases} -(Y_{dj} - Y_j)\phi'(\nu(X_j)) & \textit{if j is the output layer} \\ \delta_{j-1}W_{j-1} & \textit{otherwise} \end{cases}$$

A full derivation of the previous expressions was part of the course given by Quintero [1].

It is necessary to establish the dataset to be fed into each iteration of the backpropagation algorithm. It's worth noting that for every data point assessed within the MLP, an update of all the W_j weights in the model occurs. To address this, we propose a random partitioning of the available data as follows: we select 60% of the data for training the model after initializing the W_j weights with random $[0-1]$ values. These training data points are normalized to the [0,1] range and used sequentially (one by one) to train the model. Additionally, there is an option to use each data point multiple times during training, which is referred to as epochs. Subsequently, we evaluate the average error energy over 20% of the data and employ this value as a criteria to compare different architectures proposed in the project. Finally, we select architectures with minimum, maximum, and median values of the chosen criteria and re-evaluate the average error energy over the remaining 20% of the data, referred to as the validation set. This comprehensive approach allows us to assess the performance of various MLP configurations and make informed architectural choices.

III. Results

The initial step in conducting the proposed MLP experiments is to carefully select and segregate the respective datasets as detailed in the previous section. Figure 1 visually represents the distribution of data points and their categorization into three distinct sets: training, testing, and validation. It facilitates a transparent understanding of the dataset preparation process.

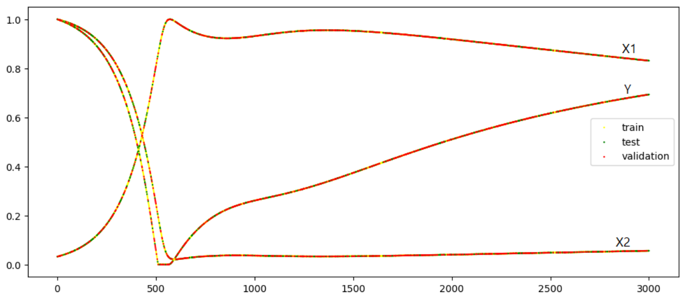

Fig. 1. Training, Test and Validation data points

Table II shows the average error energy for each of the 45 model variations describe on the previous section. The configurations (3 layers, 1 neuron per layer, $eta = 0.2$), and (3 layers, 5 neurons per layer, $eta = 0.9$), had the lowest, average and maximum error respectively. This three models were evaluated using the validation dataset and the results are presented on table I.

To further analyse the architecture's performance, we utilize a visual representation of the expected output against the output produced by three distinct models. Figure 2 was created using the test dataset.

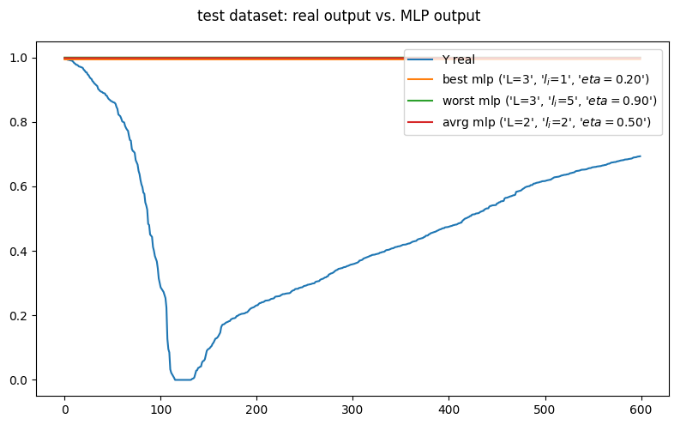

Fig. 2. Selected MLPs output againt test dataset

Considering the iterative approach employed for training the model, we have the capability to monitor the local gradients and error energy that contribute to the backpropagation algorithm. Figures 3 and 4 visually offers insights into the model's learning behavior and convergence properties.

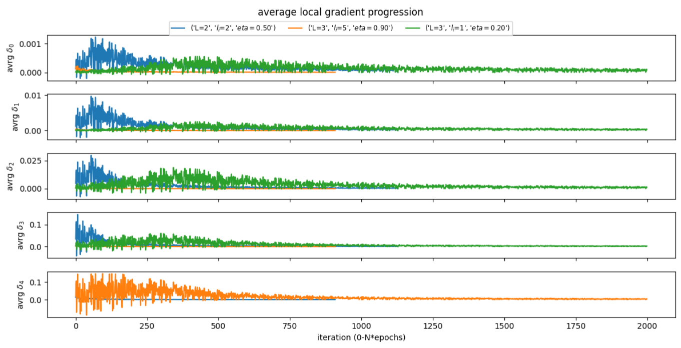

Fig. 3. Average gradient progression of selected MLPs through iterative training

Lastly, in the Appendix section, Figures 4 to 12 present a the local gradient's progression for the remaining models. These have been categorized based on the number of hidden layers and learning rates η used in their configurations.

IV. Discussion

The process of selecting the best, average, and worst-performing models was primarily guided by their average error energy when assessed on the test dataset. Nevertheless, upon closer examination, it becomes evident that their results closely resemble those of MLPs with only one layer. In other

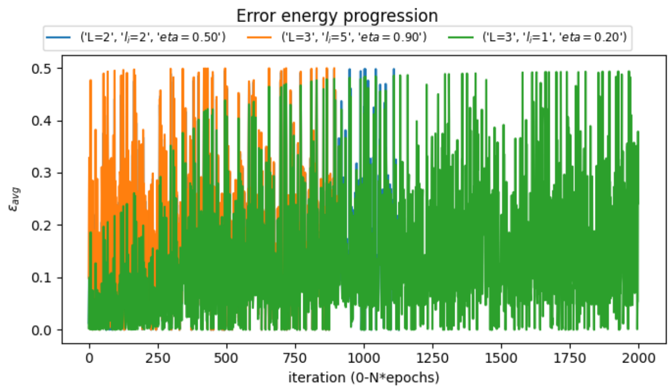

Fig. 4. Error energy progression of selected MLPs through iterative training

TABLE I
AVERAGE ERROR ENERGY ON VALIDATION DATASET

Layers	**Neurons**	η	**mean train error**	**mean test error**	**mean validation error**	**Training time [s]**
2	2	0.5	0.1543	0.1576	0.1645	42
3	5	0.9	0.1642	0.1583	0.1653	98
3	1	0.2	0.1375	0.1555	0.1623	169

words, the additional complexity introduced by more intricate architectures does not significantly enhance performance. It is plausible that a different model configuration could have been chosen as the best one. This observation underscores the limitation of evaluating a model solely based on its average error energy; which does not always provide a comprehensive assessment. Despite training more complex models, the underlying learning goal does not exhibit substantial improvement, highlighting the need for alternative evaluation metrics to better gauge model performance in certain cases.

Upon examining the outputs of the selected models on figure 2, it becomes evident that there is minimal disparity in their behavior. When attempting to fit a nonlinear line, the models seem to struggle in capturing this non-linearity effectively. The chosen activation function was a sigmoid function for all neurons. While the models do converge, this outcome is closely linked to the activation functions reaching a solution space where their behavior approximates that of a constant value of 1. In essence, these observations underscore the challenge selecting the activation functions, and their impact on the overall model behavior.

It is expected that the conclusion of the training process occurs when the gradient values stabilize. It indicates that there are no more significant updates remaining for the weights W of the MLP being trained. From our training results shown on figures 4 to 12, it is apparent that reducing the learning rate η slightly prolongs the convergence process, although no substantial differences are noticeable in the evaluation of error energy. The effect of lowering the learning rate can be interpreted as a reduction in the magnitude of weight updates. It makes sense that the algorithm would require more iterations to approach a solution space where the gradient of the error energy with respect to the weights approaches zero, given the smaller magnitude of weight adjustments. This suggests that a lower learning rate may lead to slower convergence, but the fundamental training process remains consistent.

Similar to increasing the learning rate, augmenting the number of neurons in the MLP accelerates the convergence process to some extent. Models featuring a greater number of neurons per layer introduce more variables represented by W, consequently leading to a more pronounced influence of updating the weights of the MLP during each iteration. Nonetheless, as observed in our analysis of learning rate, augmenting the number of neurons doesn't result in a substantial improvement in performance. Essentially, it translates into a more complex MLP with accelerated convergence, but without a substantial effect on achieving the learning goal.

V. Conclusion

The visual representations offer an extensive insight into the training progression of various models, categorized by their architecture. This arrangement simplifies the task of conducting a comparative analysis, enabling us to assess their learning dynamics and overall performance effectively. As we have recognized, the progression of local gradients serves as the primary indicator of a successful training process. However, the evaluation conducted on both the test and validation datasets provides valuable insights into the progression of the learning goal. Our findings reveal that despite constructing more complex MLPs, there is little improvement in the learning goal when compared to simpler models. Consequently, relying solely on the average energy error as a performance metric for ranking the models appears insufficient in capturing the full picture of their effectiveness.

Overall, the best learning outcomes were technically achieved with the models presented in the results section. Nevertheless, the computational cost associated with increasing the number of variables does not seem to be adequately justified. In future projects, we will explore additional techniques aimed at enhancing the average error energy and addressing issues such as the divergence of local gradients when employing more appropriate activation functions to accommodate nonlinearity. This signifies commitment to refining the model's performance with respect to the learning while optimizing computational efficiency in future investigations.

References

[1] Quintero, Olga Lucia. 2019. Machine Intelligence for Human Decision Making.

Appendix

TABLE II
AVERAGE ERROR ENERGY ON TRAIN AND TEST DATA SETS

Layers	η	Neurons li	Mean Train Error	Mean Test Error	Training Time [s]
1	0.9	1	0.146	0.158	36
		2	0.157	0.158	30
		3	0.158	0.158	35
		4	0.160	0.158	32
		5	0.163	0.158	33
	0.5	1	0.146	0.157	55
		2	0.157	0.158	36
		3	0.159	0.158	31
		4	0.160	0.158	36
		5	0.160	0.158	41
	0.2	1	0.141	0.156	83
		2	0.149	0.157	72
		3	0.150	0.157	69
		4	0.152	0.157	60
		5	0.158	0.157	49
2	0.9	1	0.143	0.158	50
		2	0.160	0.158	39
		3	0.161	0.158	40
		4	0.164	0.158	38
		5	0.164	0.158	51
	0.5	1	0.151	0.157	50
		2	0.154	0.158	42
		3	0.158	0.158	46
		4	0.161	0.158	50
		5	0.161	0.158	56
	0.2	1	0.136	0.156	91
		2	0.150	0.157	77
		3	0.153	0.157	85
		4	0.154	0.157	74
		5	0.157	0.157	76
3	0.9	1	0.154	0.158	43
		2	0.158	0.158	67
		3	0.161	0.158	98
		4	0.162	0.158	114
		5	0.164	0.158	98
	0.5	1	0.137	0.157	110
		2	0.153	0.158	80
		3	0.160	0.158	79
		4	0.161	0.158	85
		5	0.163	0.158	111
	0.2	1	0.137	0.155	169
		2	0.150	0.157	122
		3	0.155	0.157	123
		4	0.160	0.157	93
		5	0.158	0.157	112

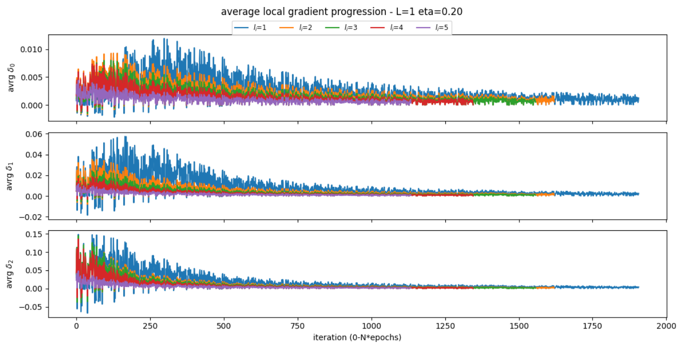

Fig. 5. Average gradient progression of MLP with L=1, $eta = 0.20$

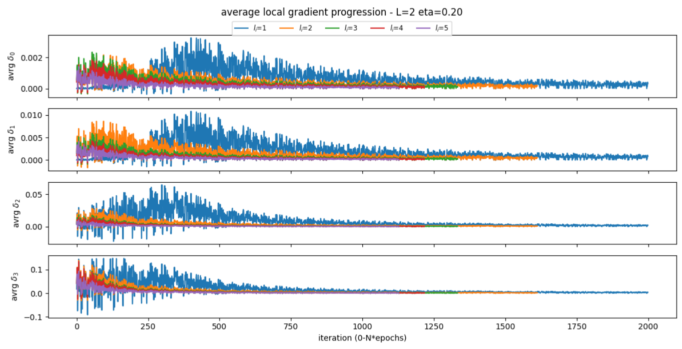

Fig. 6. Average gradient progression of MLP with L=2, $eta = 0.20$

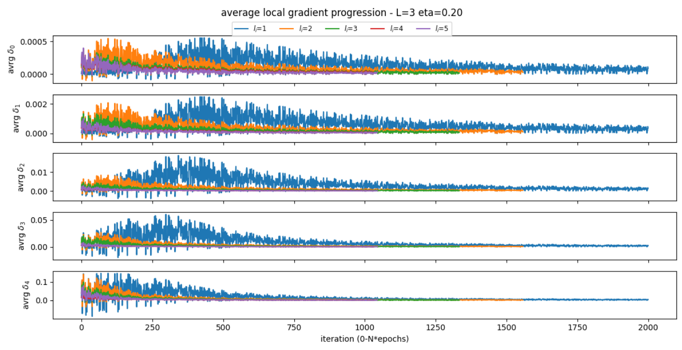

Fig. 7. Average gradient progression of MLP with L=3, $eta = 0.20$

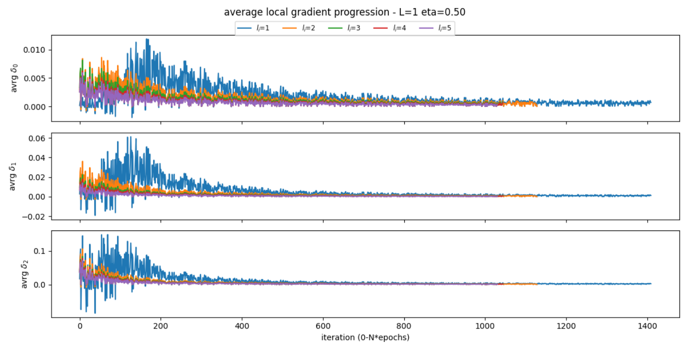

Fig. 8. Average gradient progression of MLP with L=1, $eta = 0.50$

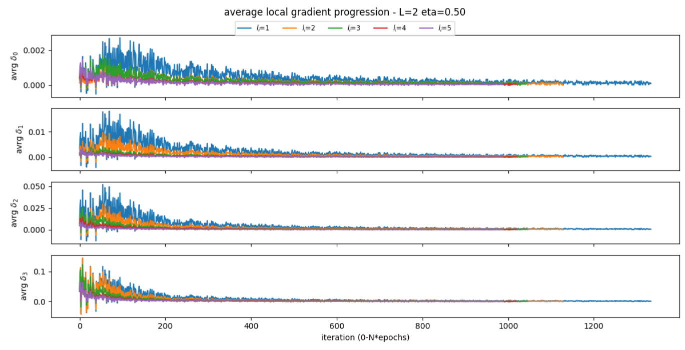

Fig. 9. Average gradient progression of MLP with L=2, $eta = 0.50$

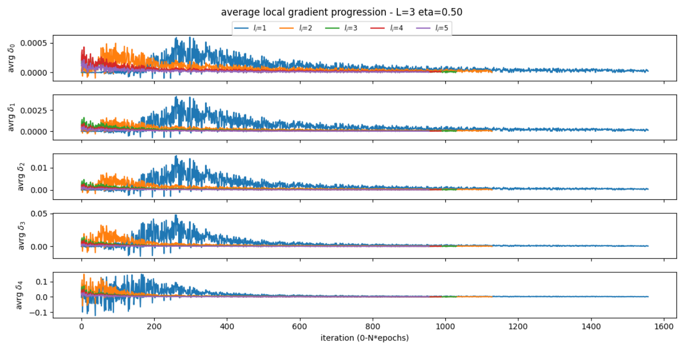

Fig. 10. Average gradient progression of MLP with L=3, $eta = 0.50$

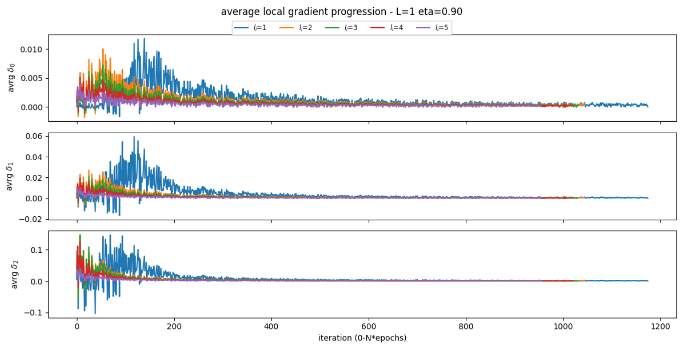

Fig. 11. Average gradient progression of MLP with L=1, $eta = 0.90$

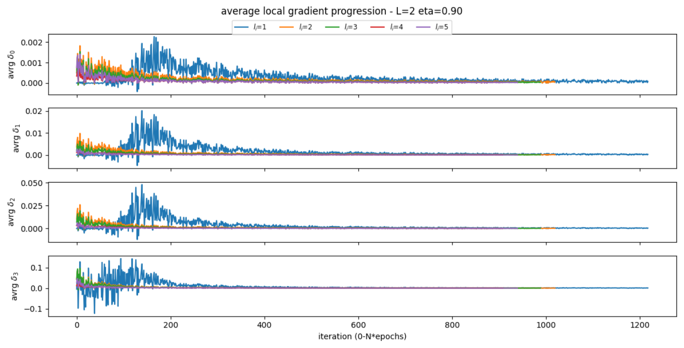

Fig. 12. Average gradient progression of MLP with L=2, $eta = 0.90$

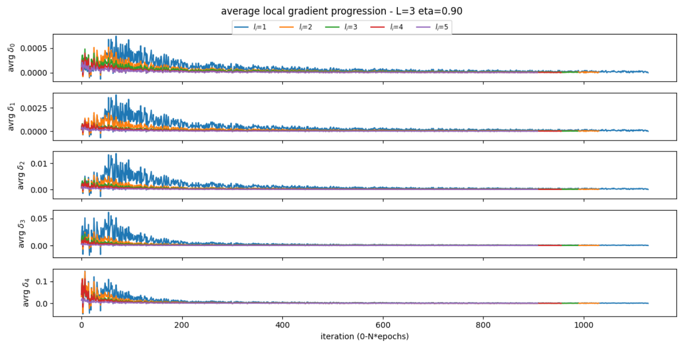

Fig. 13. Average gradient progression of MLP with L=3, $eta = 0.90$