Mini Project 2

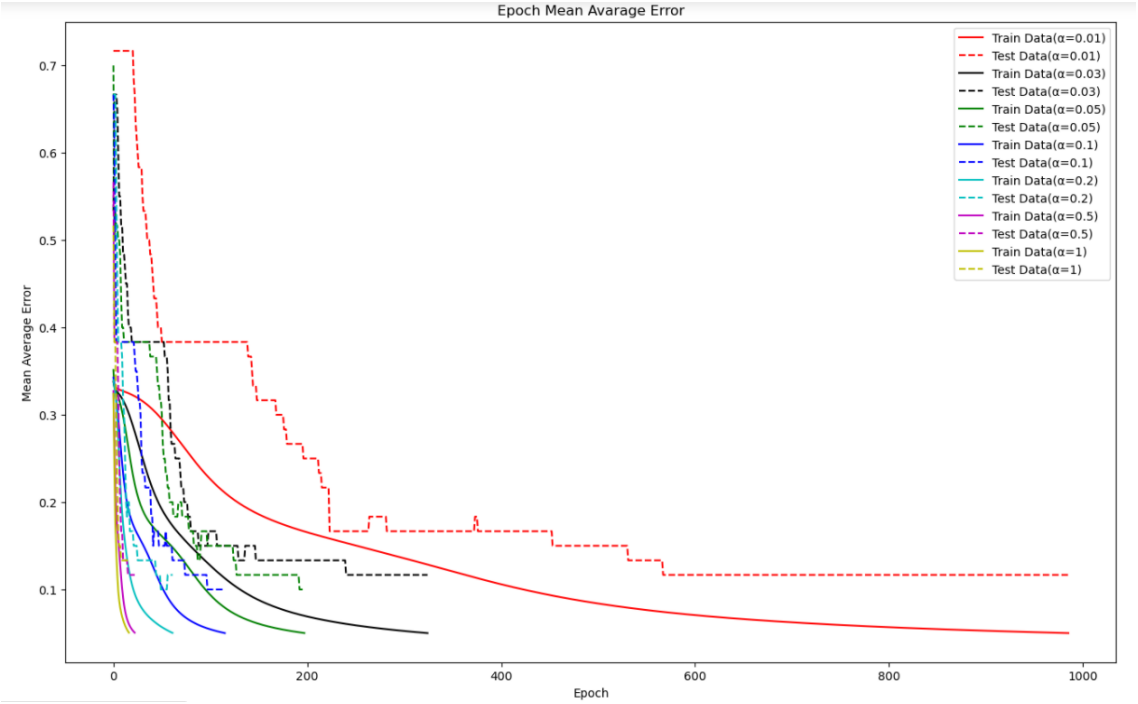
Table of Contents

Alejandro C Parra Garcia

Learning Rate

I have tried different Learning Rates of 0.01, 0.03, 0.05, 0.1, 0.2, 0.5, 1.

With them I got the next results:

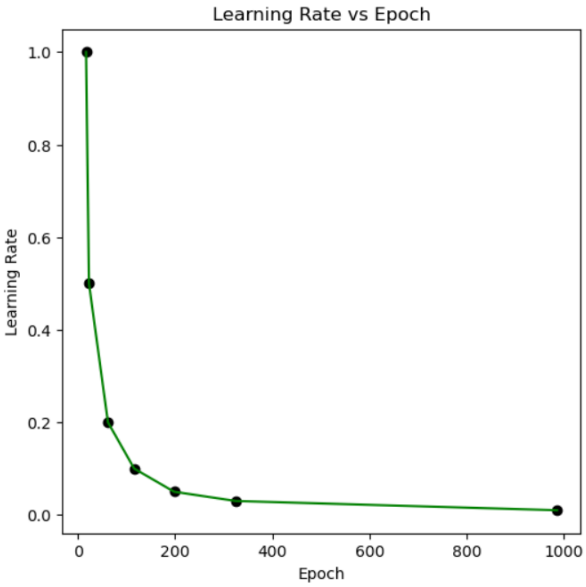


This graph represents the Average Error of the model for the Training and Test DataSet. The doted lines represent the Test data set, while the whole lines represent the Train data set.

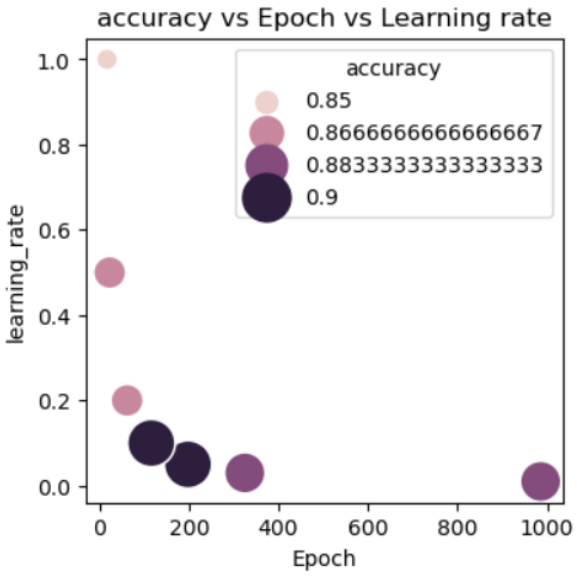
The model was train on a threshold of 0.05, and with a maximum epoch of 1.000. We can see from the graph that every model achieves the Threshold before the 1.000 epoch. But we see that each learning rate achieve it at a different rate. The slowest was the one with a learning rate of 0.01, it took 986 epochs. While the fastest one took only 17 epochs, with a learning rate of 1.00, as we can see on the next table.

	learning_rate	n_epoch	accuracy	mean_error
0	0.01	986	0.883333	0.116667
1	0.03	325	0.883333	0.116667
2	0.05	198	0.900000	0.100000
3	0.10	116	0.900000	0.100000
4	0.20	62	0.866667	0.133333
5	0.50	23	0.866667	0.133333
6	1.00	17	0.850000	0.150000

From the previous table we can see that there is a relation between the learning rate and the time it takes a model to train, so we can plot both metrics to compare them, getting the next graph:



We can see that there is an inverse relation between the learning rate and the number of epochs it took each model to train. The higher the learning rate the faster it trained. If we check for the accuracy of each model, we get the next graph:

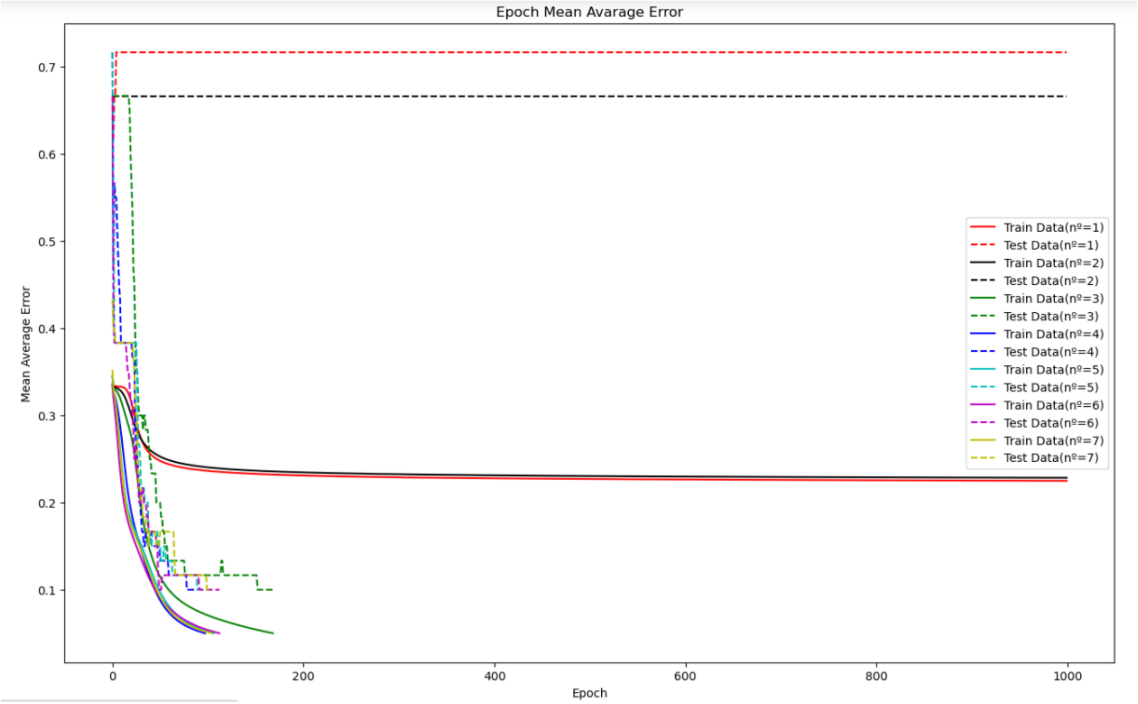


We can see that the relation between accuracy and the other metrics is lose, meaning that there isn't much relation between those variables. Although the models that seem to work the best in this case are the ones near the left down corner.

Number of Nodes in hidden layer

I have tried different number of nodes in hidden layer, from 1 to 7

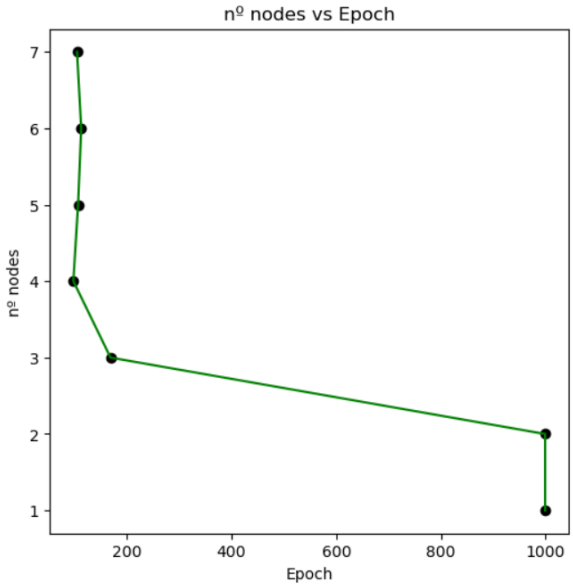
With them I got the next results:



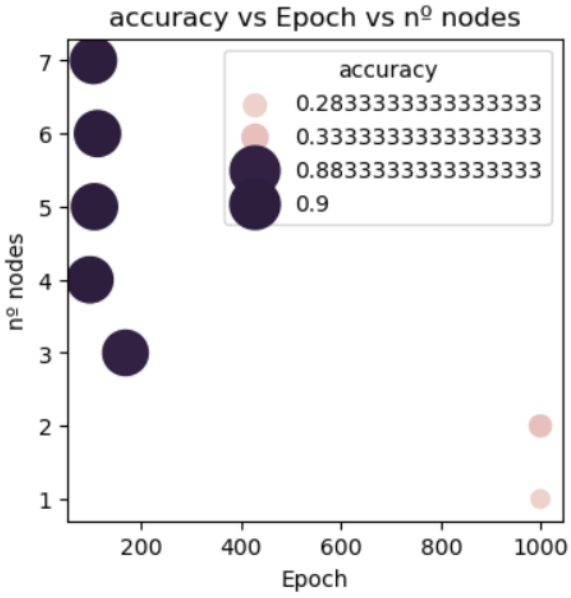
We can see from the graph that there are two distinct groups of models. One model hasn't achieved the error threshold, achieving a really bad error, in this group fall the model with 1 and 2 hidden nodes. The training dataset achieve and error between 0.2 and 0.3, while the test error was near 0.7. This means that these models didn't capture the true about the dataset. On the other group we find the rest of the models that actually achieve the error threshold and are good models.

	n_nodes	n_epoch	accuracy	mean_error
0	1	999	0.283333	0.716667
1	2	999	0.333333	0.666667
2	3	169	0.883333	0.116667
3	4	98	0.900000	0.100000
4	5	107	0.900000	0.100000
5	6	113	0.900000	0.100000
6	7	105	0.900000	0.100000

We can actually check the relation between the Number of hidden nodes and the number of epochs:



In this case we see clearly the two different models, we can see that there is no real difference between the models of 4-7 nodes, while the 3 nodes model achieve the result, but it took longer.



We can see that the 3 nodes model achieve a slightly lower accuracy that the 4-7 nodes model. And the 1-2 nodes model achieve a really low model.

So, we can see that there is a huge step between the 2 and 3 nodes model. In this case the lower node models can't capture the nature of the dataset and are unable to predict an output.

Change Activation Function

For us to try other activation Function we need to change the code in the activation function, but also on the back propagation.

First, we need the derivatives of the Relu and tanh, and the Gradient Descent:

Relu:

$$f(x) = Relu(x) = \begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x > 0 \end{cases}$$

$$\frac{d}{d\,x}\begin{cases} 0 \; if \; x \leq 0 \\ x \; if \; x > 0 \end{cases} = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x > 0 \end{cases}$$

$$Tout_k = \sigma(Wout_k) = \begin{cases} 0 \; if \; Wout_k \leq 0 \\ Wout_k \; if \; Wout_k > 0 \end{cases}$$

$$\frac{dTout_k}{dWout_k}\sigma(Wout_k) = \begin{cases} 0 \; if \; Wout_k \leq 0 \\ 1 \; if \; Wout_k > 0 \end{cases}$$

$$\begin{cases} if \; Wout_k \leq 0 \colon Tout_k = 0 \\ if \; Wout_k > 0 \colon Tout_k = Wout_k \end{cases}$$

$$\frac{dTout_k}{dWout_k}\sigma(Wout_k) = \begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases}$$

$$g(Tout_k, Wout_k) = \begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases}$$

$$g(E, W_{jk}) = g\left(E, g(Tout_k, g(Wout_k, W_{jk})\right)$$

$$\boldsymbol{g(E, W_{jk}) = (Tout_k - targ_k) * \left(\begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases}\right) * (Thid_j)}$$

$$\boldsymbol{g(E, bias_O) = \sum_k \left[(Tout_k - targ_k) * \begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases}\right]}$$

$$\text{Thid}_j = \sigma(Whid_j) = \begin{cases} 0 \; if \; Whid_j \leq 0 \\ Whid_j \; if \; Whid_j > 0 \end{cases}$$

$$\frac{d\text{Thid}_j}{dWhid_j}\sigma(Whid_j) = \begin{cases} 0 \; if \; Whid_j \leq 0 \\ 1 \; if \; Whid_j > 0 \end{cases}$$

$$g(\text{Thid}_j, Whid_j) = \begin{cases} 0 \; if \; \text{Thid}_j \leq 0 \\ 1 \; if \; \text{Thid}_j > 0 \end{cases}$$

$$\boldsymbol{g(E, w_{ij}) = \sum_k \left[(Tout_k - targ_k) * \begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases} * W_{jk}\right] * \begin{cases} 0 \; if \; \text{Thid}_j \leq 0 \\ 1 \; if \; \text{Thid}_j > 0 \end{cases} * inp_i}$$

$$\boldsymbol{(E, bias_H) = \sum_j \left[\sum_k \left[(Tout_k - targ_k) * \begin{cases} 0 \; if \; Tout_k \leq 0 \\ 1 \; if \; Tout_k > 0 \end{cases} * W_{jk}\right] * \begin{cases} 0 \; if \; \text{Thid}_j \leq 0 \\ 1 \; if \; \text{Thid}_j > 0 \end{cases}\right]}$$

Tanh:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\frac{d}{d\,x}\,\frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1}{\cosh^2(x)}$$

$$Tout_k = \sigma(Wout_k) = \frac{e^{Wout_k} - e^{-Wout_k}}{e^{Wout_k} + e^{-Wout_k}}$$

$$\frac{dTout_k}{dWout_k}\sigma(Wout_k) = \frac{1}{\cosh^2(Wout_k)}$$

$$Wout_k = tanh^{-1}(Tout_k)$$

$$\frac{dTout_k}{dWout_k}\sigma(Wout_k) = \frac{1}{\cosh^2(tanh^{-1}(Tout_k))}$$

$$g(Tout_k, Wout_k) = \frac{1}{\cosh^2(tanh^{-1}(Tout_k))}$$

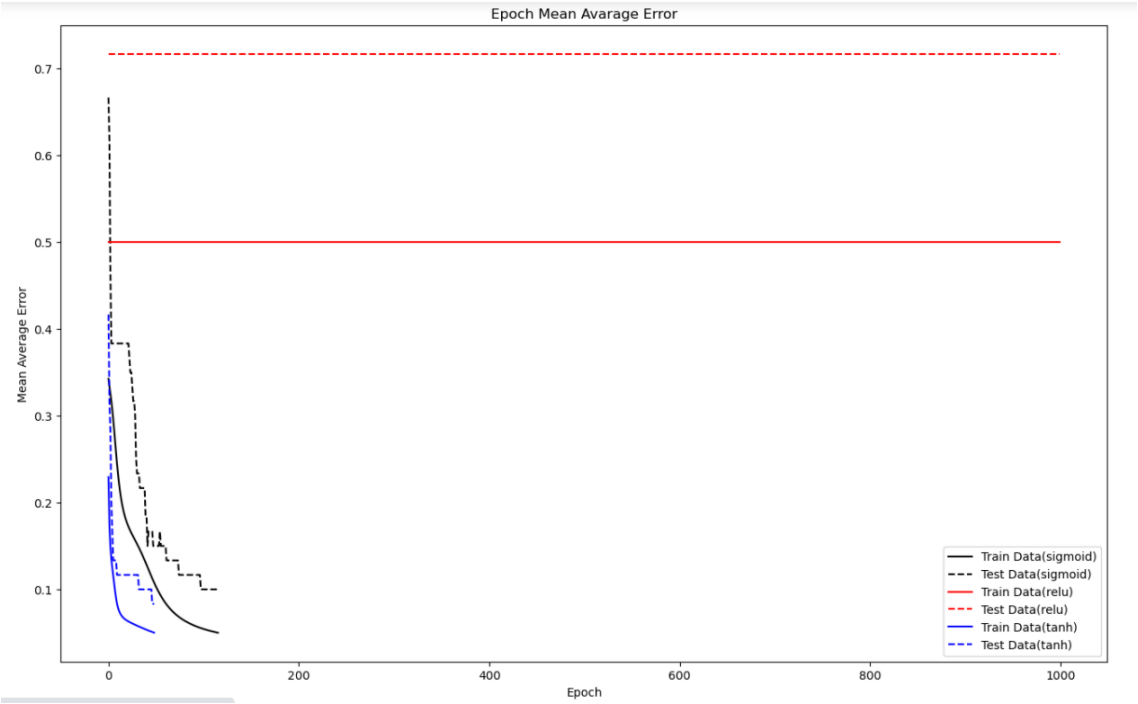
$$\boldsymbol{g(E, W_{jk}) = (Tout_k - targ_k) * \left(\frac{1}{\cosh^2(tanh^{-1}(Tout_k))}\right) * (Thid_j)}$$

$$\boldsymbol{g(E, bias_O) = \sum_k \left[(Tout_k - targ_k) * \frac{1}{\cosh^2(tanh^{-1}(Tout_k))}\right]}$$

$$\boldsymbol{g(E, w_{ij}) = \sum_k \left[(Tout_k - targ_k) * \frac{1}{\cosh^2(tanh^{-1}(Tout_k))} * W_{jk}\right] * \frac{1}{\cosh^2\left(tanh^{-1}(Whid_j)\right)} * inp_i}$$

$$\boldsymbol{(E, bias_H) = \sum_j \left[\sum_k \left[(Tout_k - targ_k) * \frac{1}{\cosh^2(tanh^{-1}(Tout_k))} * W_{jk}\right] * \frac{1}{\cosh^2(tanh^{-1}(Whid_j))}\right]}$$

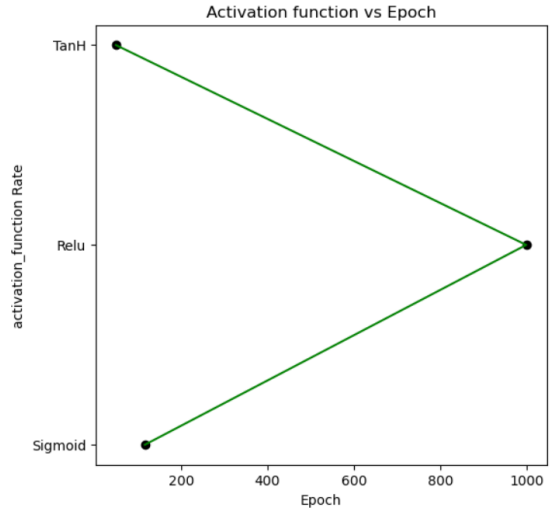
Once we have updated the functions that implemented the activation function and the back propagation, we can run the models with the new activation functions. With that we get the next results:

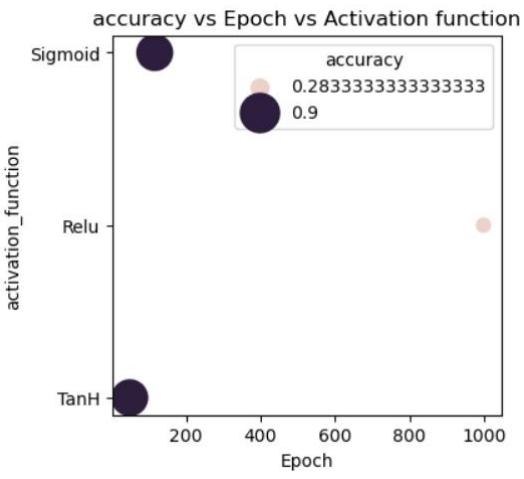


We can see that the TanH and Sigmoid activation functions reach a solution within 49 and 116 epochs respectively. On the other hand the Relu function is unable to find a solution, that can be because the output values of the Relu activation functions are [0, ∞) And thus is not a good choice for a classification problem. On the next table we find the exact numbers

	activation_function	n_epoch	accuracy	mean_error
0	Sigmoid	116	0.900000	0.100000
1	Relu	999	0.283333	0.716667
2	TanH	49	0.900000	0.100000

We can also check the other graphs:





But in this case this graph doesn’t bring much information, as we can see the same numbers as for the previous graph. The tanh and sigmoid achieve a good performance, while the Relu achieve a disastrous performance.