**Name:** Matheus Vinicius Ferreira Figueiredo Teixeira

**Student Number:** 301236904

Assignment 5 – Neural Network

# Exercise 1

## Introduction

For this exercise we were asked to generate two columns of input data, and one column of output. The data set should have 10 rows and the output should be the sum of the inputs, following the equation:

Which can also be written as:

If we plot this equation in a 3D graphic, we will see a plane like this:

Chart, radar chart

Description automatically generated

Figure 1: Plot of f(x1, x2) = x1 + x2, generated using the online mathematical tool wolfram alpha.

The initial data generated for this exercise is presented in the following table just as an example; for the other exercises, this kind of table will not be presented.

Table 1: Data used for exercise 1.

|  |  |  |
| --- | --- | --- |
| **x1** | **x2** | **output** |
| -0.0995736 | -0.0969666 | -0.19654 |
| 0.264389 | 0.222263 | 0.486653 |
| -0.599863 | -0.354657 | -0.95452 |
| -0.237201 | 0.453741 | 0.21654 |
| -0.423893 | -0.567135 | -0.991028 |
| -0.489194 | 0.204561 | -0.284633 |
| -0.376488 | -0.0992342 | -0.475722 |
| -0.185327 | 0.0704278 | -0.114899 |
| -0.123879 | -0.431536 | -0.555415 |
| 0.0465801 | -0.362278 | -0.315698 |

After generating the data, it was asked to generate a neural network with 6 neurons and train this neural network with the above data. This will be the standard; the other exercises will be variations of it, so one can make comparisons and analysis.

# Exercise 2

## In your written response compare the result #1 to result #2 to the actual result explain your findings.

It was asked to pass the input (0.1, 0.2) to the trained networks. In addition to that input, I have also passed the input (0.6, 0.6) to enrich the analysis. The results are summarized in the following table.

Table 2: Summary of exercise 1 and 2.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Exercise 1** | **Exercise 2** |
|  | Number of Epochs | **90** | **1000** |
|  | Average Error | **6.74 \* 10-6** | **1.45\*10-2** |
|  | Train size | 10 | 10 |
|  | Number of layers | 1 | 2 |
| Data input: (0.1, 0.2) | Network prediction | 0.2174 | 0.3295 |
| Expected Value | 0.3000 | 0.3000 |
| Error | **0.0826** | **-0.0295** |
| Data input: (0.6, 0.6) | Network prediction | 1.0000 | -0.2775 |
| Expected Value | 1.2000 | 1.2000 |
| Error | **0.2000** | **1.4775** |

Comparing the results from the neural networks created in exercises 1 and 2, one can notice that increasing the number of layers made the network take more time to converge. It also increased the average error by a factor of 104 (10,000 times). For the region near the training data – such as the region of the input (0.1, 0.2) – increasing the number of layers improved the prediction; it made the error (expected value minus predicted value) drop from 0.0826 to -0.0295. On the other hand, for regions more distant from where the training data is concentrated – such as the region of the input (0.6, 0.6) - the error increased considerably from 0.2 to 1.48.

The conclusion from the above analysis is that increasing the number of layers can cause overfitting if you have too little data. Increasing the number of layers will make improvements only in the predictions for inputs close to the training data, but it may cause the overall error to increase, this is a symptom of overfitting. This conclusion is backed up by the graphics presented in figures 2 and 3.

On figures 2 and 3, the gray-ish plane is the plot of the original equation (); the orange dots represent the predictions for the original inputs (presented in table 1); and the blue line is a set of an additional 600 predictions made by each trained neural network. One can notice that despite x1 and x2 being generated by the uniform distribution, most of the inputs and outputs got concentrated on the negative region of the plot; it has 8 data points in the down part of the plane and only 2 on the up part. That happened because the numbers were generated randomly and 10 is a very low number of points, which is not enough to guarantee that they will be evenly distributed over the entire region. That being said, it is very easy to notice that increasing the number of layers slightly improved the predictions for the down region of the plane; however, the predictions for the upside of the plane were drastically worsened.

Chart, surface chart

Description automatically generated

Figure 2: The plot from exercise 1. The gray region represents the original equation plot (output = x1 + x2). The orange dots are the predictions for the original 10 data inputs. The blue line is the prediction for additional 600 inputs in the plane diagonal. The network has a single layer with 6 neurons.

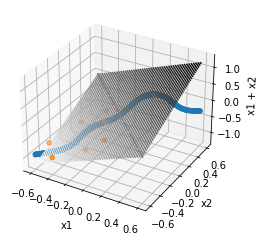


Figure 3: The plot from exercise 2. The gray region represents the original equation plot (output = x1 + x2). The orange dots are the predictions for the original 10 data inputs. The blue line is the prediction for additional 600 inputs in the plane diagonal. The network has 2 layers: the first with 5 neurons, the second with 3 neurons

# Exercise 3

## In your written response compare the result #1 to result #2 to the actual result explain your findings.

Table 3: Summary of exercise 1 and 3.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Exercise 1** | **Exercise 3** |
|  | Number of Epochs | **90** | **495** |
|  | Average Error | **6.74 \* 10-6** | **9.24\*10-3** |
|  | Train size | 10 | 100 |
|  | Number of layers | 1 | 1 |
| Data input: (0.1, 0.2) | Network prediction | 0.2174 | 0.298 |
| Expected Value | 0.3000 | 0.3000 |
| Error | **0.0826** | **0.0020** |
| Data input: (0.6, 0.6) | Network prediction | 1.0000 | 0.9960 |
| Expected Value | 1.2000 | 1.2000 |
| Error | **0.2000** | **0.2040** |

Looking at table 3, one can notice that increasing the number of data makes a significant difference. Despite the increment in the number of epochs and the average error, the latest one is still very small. The error for the (0.1, 0.2) input dropped about 98%, and the error for the (0.6, 0.6) input is almost the same in both cases. This is an indication of how important is to have a reasonable number of data points to train.

Looking at the visual plot (figure 4), one can notice how the training data is much better distributed on the plane. Also, the predictions for the additional 600 inputs are almost a straight line sitting right in the plane.

Chart, surface chart

Description automatically generated

Figure 4: The plot from exercise 1. The gray region represents the original equation plot (output = x1 + x2). The orange dots are the predictions for the original 100 data inputs. The blue line is the prediction for additional 600 inputs in the plane diagonal. The network has a single layer with 6 neurons.

# Exercise 4

## Plot the error training size graph

Chart, line chart

Description automatically generated

Figure 5: Error size graph for exercise 4.

## In your written response compare the result #3 to result #4 to the actual result, explain your findings

Table 4: Summary of exercise 3 and 4.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Exercise 3** | **Exercise 4** |
|  | Number of Epochs | **495** | **1000** |
|  | Average Error | **9.24\*10-3** | **2.93\*10-1** |
|  | Train size | 100 | 100 |
|  | Number of layers | 1 | 2 |
| Data input: (0.1, 0.2) | Network prediction | 0.298 | 0.2743 |
| Expected Value | 0.3000 | 0.3000 |
| Error | **0.0020** | **0.0257** |
| Data input: (0.6, 0.6) | Network prediction | 0.9960 | 0.6160 |
| Expected Value | 1.2000 | 1.2000 |
| Error | **0.2040** | **0.5840** |

One can look at table 4 and figure 6 and get the impression that increasing the number of layers only made things worse. However, that would be a hasty conclusion. What really happened is that, on exercise 4, the neural network started to get big. Because it has 100 data inputs, 8 neurons, and 2 layers, it started to take much more time to converge. As one can notice from the graphic presented in figure 5, after 60 epochs or so, the drop in the training error really slows downs; it means that many epochs are needed to make the error drops in small amounts. The training was set to do 1,000 epochs at maximum, but 1,000 epochs were not enough to make the error drop to the level obtained in exercise 3. To confirm that, for the sake of the completeness of this analysis, and my curiosity, I re-ran exercise 4 but allowed it to run 50,000 epochs. What I could see is that the error was continuously dropping. At epoch 50,000 it reached 3.66 \* 10-2. I have made the 3D plot for the 50,000 epoch train as a courtesy (figure 7).

Chart, surface chart

Description automatically generated

Figure 6: The plot from exercise 4. The gray region represents the original equation plot (output = x1 + x2). The orange dots are the predictions for the original 100 data inputs. The blue line is the prediction for additional 600 inputs in the plane diagonal. The network has 2 layers: the first with 5 neurons, the second with 3 neurons.

Chart, surface chart

Description automatically generated

Figure 7: Re-run of exercise 4. The result of letting the neural network train for 50,000 epochs (a courtesy of Matheus’ curiosity).

# Exercise 5

## In your written response compare the result #5 to result #6 to the actual result, explain your findings.

Table 5: Summary of exercise 5.9 and 5.11.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Exercise 5.9** | **Exercise 5.11** |
|  | Number of Epochs | **225** | **1000** |
|  | Average Error | **1.71\*10-2** | **7.37\*10-1** |
|  | Train size | 10 | 100 |
|  | Number of layers | 1 | 2 |
| Data input: (0.2, 0.1, 0.2) | Network prediction | 1.0000 | 0.5111 |
| Expected Value | 0.5000 | 0.5000 |
| Error | **-0.5000** | **-0.0111** |
| Data input: (0.6, 0.6, 0.6) | Network prediction | 1.0000 | 0.9911 |
| Expected Value | 1.8000 | 1.8000 |
| Error | **0.8000** | **0.8089** |

For this exercise, we have four variables (three inputs and one output); it is a 4D problem, so we cannot plot a 3D graphic to have a nice visualization; therefore, this analysis will rely on table 5 only. Now, with three features as input, even the simplest network (the one with ten training data and a single layer with six neurons) takes 225 epochs to converge. This is a 150% increase if compared to the number of epochs taken in exercise 1. One can conclude from this that the neural network takes now much more runs to converge.

If one compares the results of item 9 and item 11 of exercise 5, he will notice that for the input (0.2, 0.1, 0.2) the error was hugely improved. It dropped from -0.5000 to -0.0111; this is the expected behavior, for we increased the number of training data from 10 to 100, the number of layers from 1 to 2, and the number of neurons from 6 to 8. The error did not change much for the input (0.6, 0.6, 0.6). Despite these observations, one can notice that the average error is a little higher on item 11 than it was on item 9. Similarly to what happened in Exercise 4, this effect is caused because we set the maximum number of epochs to 1,000, and this neural network became too big to be processed in only 1,000 epochs. In item 11, the neural network has three features, 100 training data, two layers, and eight neurons, and because of that, the error drops very slowly, as we can notice in figure 8. If we free the number of epochs to be 50,000, for example, the overall error average will probably drop.

Table 6: Error size graph for exercise 5.

Chart, line chart

Description automatically generated

# Conclusion

From the exercises related above and their corresponding analysis, on get some conclusions. First, in general, increasing the number of layers (and neurons) tends to improve the neural network; however, it may take much more epochs before converging. Second, increasing only the number of layers without taking into consideration the size of the training data, can cause overfitting; having a proper size for the training data is crucial to guarantee that the entire region of interest will be represented by that training data; on the other hand, having too many training data can considerably increase the number of runs to make the neural network to converge. Finally, increasing the number of features also increases the number of runs to make the neural network converge.