SHPC4001 - Assignment 7

Ayden S. McCann

April 2021

Code Listing at the end

Introduction The purpose of this assignment was to gain a familiarity with how neural networks operate. In question 1 a neural network has been constructed by hand, with weights and biases manually chosen to act as an XOR gate. For comparison, in question 2, a more complete neural network based on code obtained from [1] was used in conjunction with a personally implemented back propagation algorithm for an N layer neural network. After training this code on an XOR gate, it was then applied to a real physics application, namely trying to predict failures of O-rings in the Space Shuttle Challenger's SRBs.

1 Question 1

An Artificial Neural Network (ANN) was designed with the following structure:

x_1, x_2 inputs $\rightarrow$ 2 nodes in a hidden layer $\rightarrow$ 1 output (see Figure 1). Where the activation functions are Sigmoid (1), and Identity ($x = x$)

$$f(x) = \frac{1}{1 + e^{-}x} \tag{1}$$

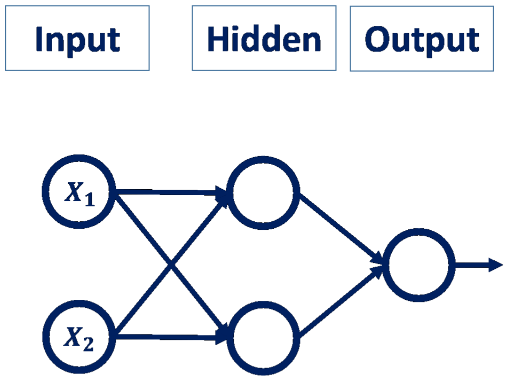
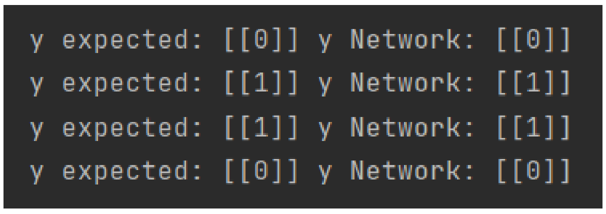


Figure 1: A Diagram representing the structure of the ANN in Q1.py for Question 1.

With the choice of given weights and biases in Q1.py the expected outputs of $1 \approx 1$, while $0 \approx -0.5, 1.8$. While this is not the ideal output it is clear to see which inputs are intended as outputs of 1 as they are within 0.023 of 1. This can be further clarified by rounding the outputs then performing an if / else loop to allocate $1 = 1$ else 0. The outputs after performing such an operation is shown in Figure 2.



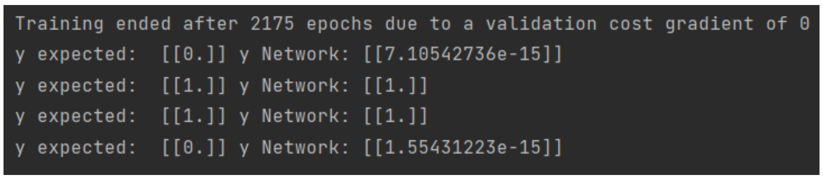
```
y expected: [[0]] y Network: [[0]]
y expected: [[1]] y Network: [[1]]
y expected: [[1]] y Network: [[1]]
y expected: [[0]] y Network: [[0]]
```

Figure 2: The results from the ANN in Q1.py for Question 1.

2 Question 2

The difference between the code utilised in this section and that from Q1 is the inclusion of the back propagation algorithm over a number of epochs and batches. The procedure in question 1 was to manually define the weights and biases. In the Q2 code, the weights are updated through the use of learningRate, weightsGradients and batchSize while the biases are updated using learningRate deltaSum and batchSize.

The back propagation algorithm defined in Q2.py is generalised for an N layer ANN using for loops. An additional inclusion featured in the supplied was a break in the epoch loop if the validation cost doesn't change between layers. I included this as i noticed once the code had been iterating for a long period of time it would reach a point where both validation cost and training cost would both reach a constant value. Iterating further from this point is needless hence the break in the loop. The following results were obtained a learning rate of 0.3 and 10,000 epochs. However the training ended after 2175 epochs due to the validation cost break as previously mentioned.



```
Training ended after 2175 epochs due to a validation cost gradient of 0
y expected:  [[0.]] y Network: [[7.10542736e-15]]
y expected:  [[1.]] y Network: [[1.]]
y expected:  [[1.]] y Network: [[1.]]
y expected:  [[0.]] y Network: [[1.55431223e-15]]
```

Figure 3: The results from the ANN in Q2.py for Question 2. Note the printed message remarking the early cessation of the epoch loop.

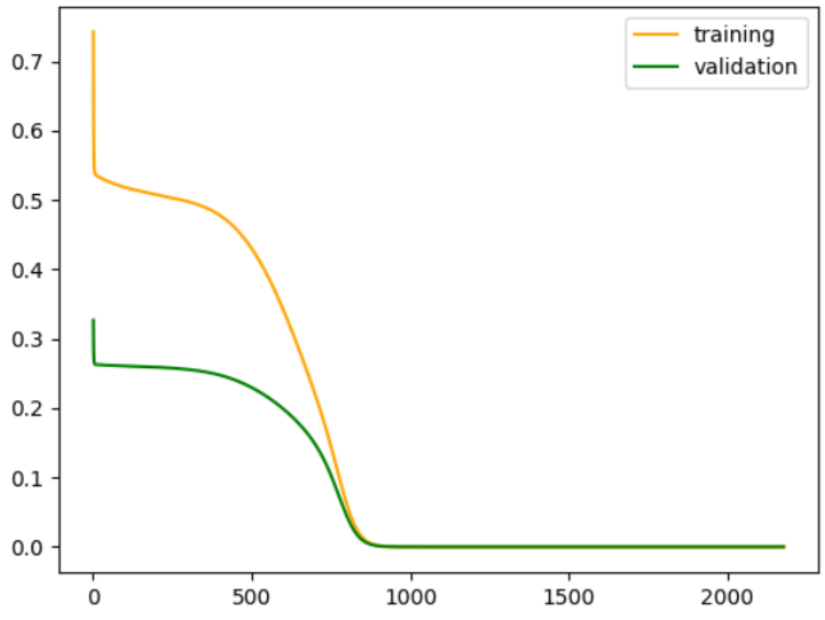


Figure 4: The plot of training and validation costs per epoch from Q2.

3 Question 3

The dataset chosen for this question contained values for the Space Shuttle Challenger's O-rings [2]. A failure in one of these O-rings due to unusually low temperatures at launch resulted in the Challenger disaster in 1986 [3]. The purpose of training on this dataset is to predict the number of O-rings experiencing thermal distress. The data consists of 23 rows formatted in the following way:

1. Number of O-rings at risk on a given flight
2. Number experiencing thermal distress
3. Launch temperature (degrees F)
4. Leak-check pressure (psi)
5. Temporal order of flight

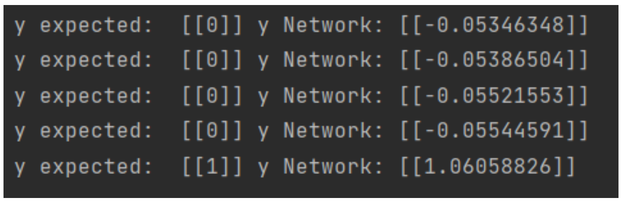
Where the integers represent columns separated by whitespace. Columns 1,3,4,5 are the X inputs, while Column 2 is the desired prediction by the system.

The layout of the ANN applied to this problem was the following: 4 inputs, a 100 neuron layer, a 200 neuron layer, another 200 neuron layer followed by 1 output.

The 3 hidden layers all had sigmoid activation functions and the output layer simply had $x = x$ (Identity) as its activation function.

The first 18 sets of data were used for training while the final 5 were used as validation ($\sim 80 : 20$).

The following results were obtained with 10,000 iterations and a learning rate of 0.01 (sufficient results can be obtained with 1,000).



```
y expected:  [[0]] y Network: [[-0.05346348]]
y expected:  [[0]] y Network: [[-0.05386504]]
y expected:  [[0]] y Network: [[-0.05521553]]
y expected:  [[0]] y Network: [[-0.05544591]]
y expected:  [[1]] y Network: [[1.06058826]]
```

Figure 5: The results from the ANN in Q3.py for Question 3.

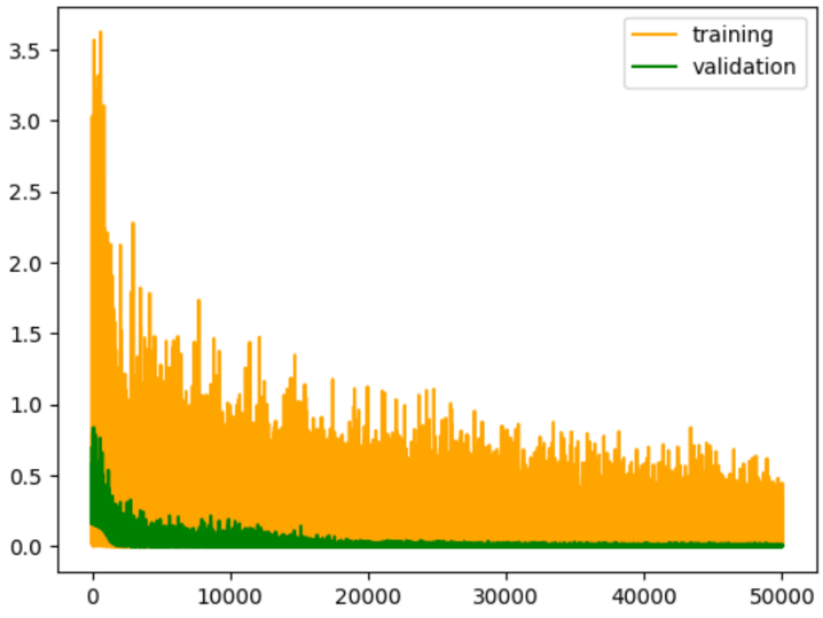


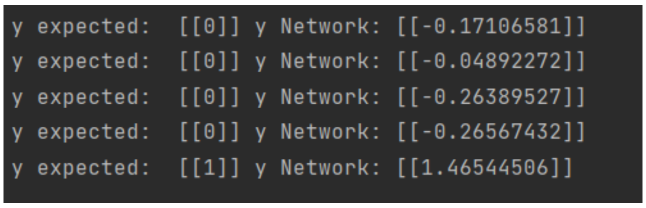
Figure 6: The plot of training and validation costs per epoch from Q3.

Clearly this more complex network with 4 inputs was a lot more difficult for the network to accurately predict. The validation costs seem to oscillate between a base level which is quite defined and a more unpredictable upper limit. This lower value progresses from roughly 0.2 down to 0. I believe the discrepancy between the training and validation costs is due purely to the difference in the numbers of data sets inputted into each. Given the validation set only contains 4 rows of data it is less difficult for the network to predict. It begins with a lower cost and it also converges closer to zero in a lot sorter time. The noise like fluctuations in both costs are due to the different combinations of input variables being shuffled and thus each having largely different costs until the network becomes very accurate.

Because this more complicated network takes a lot longer to compute, the readout of training and validation costs was replaced with a print out of the epoch, so as to keep track of the progress.

3.1 Additional testing

After observing the training cost plots exhibiting a low but evident convergence in Figure 6 I decided to increase the number of epochs to 100,000 (which took 5 hours) to establish whether or not the accuracy increased. Unfortunately this was not the case, while the training cost did indeed converge closer to zero the validation cost ended up shifting to a fairly consistent value of ~ 0.15. This resulted in less accurate predictions for the validation set. This is perhaps evidence of over-training where the system begins training on statistical noise rather than components of the physical system. Given that the training set with 18 pieces of data did not encounter this perhaps the 5 validation sets were too brief to employ such a large number of epochs without having this behaviour hinder its' predictions.



```
y expected:  [[0]] y Network: [[-0.17106581]]
y expected:  [[0]] y Network: [[-0.04892272]]
y expected:  [[0]] y Network: [[-0.26389527]]
y expected:  [[0]] y Network: [[-0.26567432]]
y expected:  [[1]] y Network: [[1.46544506]]
```

Figure 7: The results of the 100k epoch training.

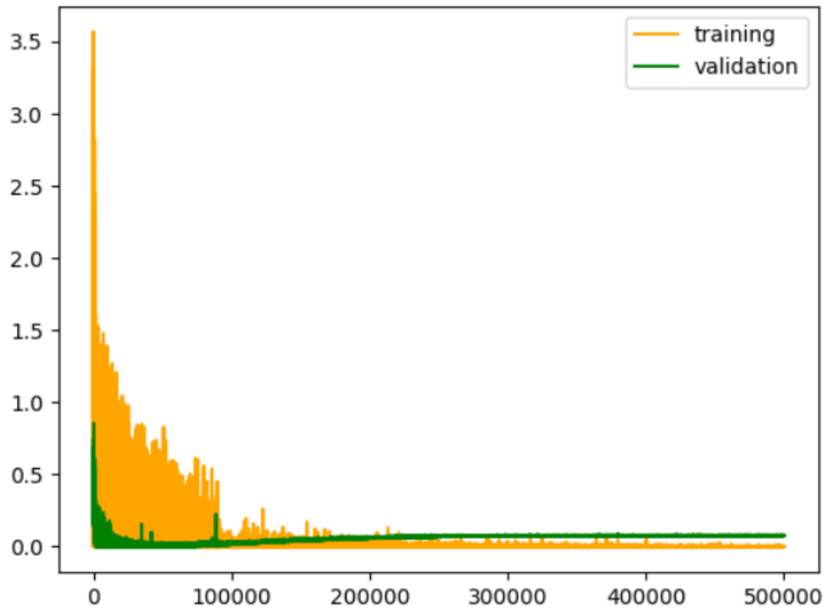


Figure 8: The plot of training and validation costs for the 100k training.

Conclusion Neural networks are incredibly diverse and having applications in a range of fields. The back propagation algorithm is a very important component of these networks and allow them to make detailed predictions that would simply be impossible to manually configure.

Note to marker

I have made this report fairly shorter than others on the advice given to us during the workshop.

4 Code Listing

Question 1: Q1.py

Question 2: Q2.py

Question 3: Q3.py

References

[1] Matwiejew E, 2021, Edric-Matwiejew/myANN. `https://github.com/Edric-Matwiejew/myANN`

[2] Challenger USA Space Shuttle O-Ring Data Set, UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Challenger+USA+Space+Shuttle+O-Ring`

[3] Rogers Commission Report, 1986, Report of the Presidential Commission on the Space Shuttle Challenger Accident `https://www.govinfo.gov/content/pkg/GPO-CRPT-99hrpt1016/pdf/GPO-CRPT-99hrpt1016.pdf`