# Research/Programming Assignment

Lukasz Filanowski 18414616

## Code and Project Overview

The assignment was written in Python. The code portion of the assignment is divided into 3 main files. Multi_Layered_Perceptron.py contains the core functionality of the MLP. XOR.py contains the testing and training done for the XOR example using a variety of Learning Rate values and Hidden Layer count. Similarly, the Sin.py file contains testing and training for the Sin example. This too has been tested for a variety of Hidden Layer and Learning rate values. In order to run these examples, simply run either the XOR.py file or Sin.py file. Each parameter for the MLP can be changed via the variables at the top of each file. The output generated by each program is outputted to a generated txt file, who's name is the name of the program followed by Learning Rate and Hidden Layer values. For example, the name generated for the example on the left would be XOR_0.2_4.txt. The Sin example takes a bit of time to run (around 3-4 minutes). The main method of storing values in this assignment is using arrays. Initially, efforts were made to use numpy arrays, but these proved too inconsistent and didn't follow standard syntax rules.

```
EPOCHS = 10000
LEARNING_RATE = 0.2
INPUTS = 2
HIDDEN = 4
OUTPUTS = 1
```

The files produced from experimentation with XOR and SIN are included in the submission.

## XOR Results

The XOR example was trained using the sigmoidal function on a forward pass. The weights were updated on each epoch. This training and testing was repeated for a variety of different Hidden Layer and Learning Rate values. For each of these, the **Average Difference** between the Target and Actual Output (**average of how much the result differed from expected value**) was calculated and displayed at the bottom of the file. The error was displayed every 200 Epochs out of a total of 10000. The results can be seen below. The lower the Average Difference, the closer the MLP was to predicting correctly.

| Hidden Layers | Learning Rate | Error at Epoch 0 | Error at Epoch 9800 | Error Difference | Average Difference |
|---|---|---|---|---|---|
| 3 | 0.1 | 0.4946 | 0.0883 | 0.4063 | 0.0866 |
| 4 | 0.1 | 0.4997 | 0.0991 | 0.4006 | 0.0965 |
| 5 | 0.1 | 0.4979 | 0.0797 | 0.4182 | 0.0780 |
| 4 | 0.01 | 0.4967 | 0.4977 | -0.001 | 0.4974 |
| 4 | 0.2 | 0.4985 | 0.0536 | 0.4149 | 0.0527 |
| 4 | 0.5 | 0.5018 | 0.0307 | 0.4711 | 0.0302 |
| 4 | 0.7 | 0.4993 | 0.0253 | 0.474 | 0.0249 |

As the results demonstrate, the increase in hidden layers has small impact on the error difference and average difference of results. Interestingly, at 4 Hidden Layers, the error increases significantly and the difference increases. With HL (Hidden Layers) = 5 and LR (Learning Rate) = 0.1 the error decreased and the overall difference decreases, fitting back into the trend. It was decided to investigate the effect of altering the Learning Rate as Hidden Layers seemed to provide little to no major change. We first investigate decreasing LR to 0.01 and keeping HL at 4. The aim of this was to try to improve the accuracy for this outlier in the previous set of tests. This showed an overall increase in error between first and last epoch as well as a very high average difference (0.4974). The LR was then increased

gradually and with each iteration the error decreases and the overall average difference decreases gradually. This indicates an improvement in accuracy of the model with higher Learning Rate and consistent Hidden Layers.

Overall, the model was able to reasonably accurately predict the values with very low error values and low average difference values, indicating a small difference between the predicted and target values.

Results for best achieved run (HL = 4, LR = 0.7) and Average Difference values:

```
Target: [0] Output: [0.027697210929143533]
Difference: [0.02769721]

Target: [1]    Output: [0.9779071758143392]
Difference: [0.02209282]

Target: [1]    Output: [0.9745906009713882]
Difference: [0.0254094]

Target: [0]    Output: [0.024753616521730956]
Difference: [0.02475362]

Average difference: 0.02498826266628679
```

## SIN Results

The Sin example was tested using hyperbolic tangent and forward(). The input was divided among 2 input arrays: sin_input and sin_test. Sin_input contained 400 inputs that were calculated using 4 randomly generated numbers in an array between -1 and 1. The numbers were the manipulated using the formula: sin(x1-x2+x3-x4) and that result had the sin function applied to it and was saved into the corresponding desired output arrays. Similarly to the XOR example, the weights were updated at every epoch and the average difference was calculated for both the sin_input results and the regular sin_test results which were not trained on. These tests were ran for different LR and HL values and the result can be seen below:

| Hidden Layers | Learning Rate | Error at Epoch 0 | Error at Epoch 9800 | Error Difference | Average Difference (sin_input) | Average Difference (sin_test) |
|---|---|---|---|---|---|---|
| 5 | 0.1 | 0.2085 | 0.1037 | 0.1048 | 0.1069 | 0.1208 |
| 6 | 0.1 | 0.1960 | 0.0988 | 0.0972 | 0.0875 | 0.0611 |
| 7 | 0.1 | 0.1748 | 0.1029 | 0.0719 | 0.1043 | 0.1115 |
| 5 | 0.01 | 0.5841 | 0.0642 | 0.5199 | 0.0751 | 0.0783 |
| 5 | 0.2 | 0.1763 | 0.1009 | 0.0754 | 0.0914 | 0.0920 |
| 5 | 0.5 | 0.1212 | 0.1199 | 0.0013 | 0.1074 | 0.1057 |
| 6 | 0.01 | 0.5462 | 0.0852 | 0.461 | 0.0848 | 0.0820 |

We can see from the results that increasing the HL with a constant LR increases the accuracy of the model both in terms of decreasing error and decreasing average difference. In most cases the sin_input performed better than the sin_test except in the case of HL = 6 and LR = 0.1 where a significant difference can be seen. This can be deemed as an outlier. The effects of changing the LR were then investigated. A low LR of 0.01 produced a significant error spike which dropped significantly in the later epochs. A significant improvement can be seen in the average differences which yield very low values. Further increasing the LR only decreased the accuracy of the model average differences. However, the initial error value is reduced compared to previous tests. A final test was run to investigate a seemingly better low LR and increased HL. This produced a high initial error which decreased but not as low as the best run of HL = 5 and LR = 0.01. The average differences weren't as good as this best run either. It appears that a low LR yields the best results however and HL should be adjusted to find the optimal combination.

Sample of results for best run (HL = 5, LR = 0.01) and Average Difference values:

```
Target: [0.7008499906546477]    Output: [0.737898557962999]
Difference: [0.03704857]

Target: [-0.1589668437742307]  Output: [-0.13946809067750784]
Difference: [0.01949875]

Target: [0.6252338857168102]   Output: [0.7312061437910531]
Difference: [0.10597226]

Target: [0.9574563916660913]   Output: [0.9010006174955767]
Difference: [0.05645577]

Target: [0.3254462788474712]   Output: [0.31611136835154524]
Difference: [0.00933491]

Target: [0.5002564456780668]   Output: [0.42021072834715256]
Difference: [0.08004572]

Target: [-0.9998033379832035]  Output: [-0.9434861091499387]
Difference: [0.05631723]

Target: [-0.41668699252507674] Output: [-0.5894412092267642]
Difference: [0.17275422]

Target: [0.9683327053674781]   Output: [0.8627964449731448]
Difference: [0.10553626]

Target: [-0.13025211531071396] Output: [-0.07429948394809587]
Difference: [0.05595263]

Average Difference for sin_input: 0.07513559347778193
Average Difference for sin_test: 0.07832486957324
```

## Conclusions

Based on the results obtained, the Learning Rate has the biggest influence on the model's accuracy and error margin. This applies to both the XOR and SIN examples. For XOR, a Hidden Layer count of 4/5 is optimal with a high Learning Rate. However, in the case of SIN the opposite was true as a low Learning Rate yielded the best results with the Hidden Layers remaining roughly the same. The assignment allowed me to get a better understanding of the inner workings of a Multi Layered Perceptron. It also gave me an appreciation of the work that goes behind designing and implementing a Neural Network learning model.