Artificial Intelligence CMPU3051: 2020 - 21

Assignment 1 MLP

Submission Date:18/12/2020

Lecturer Name: Richard Lawlor

Student

Philip Herweling

C18470774

C18470774@mytudublin.ie

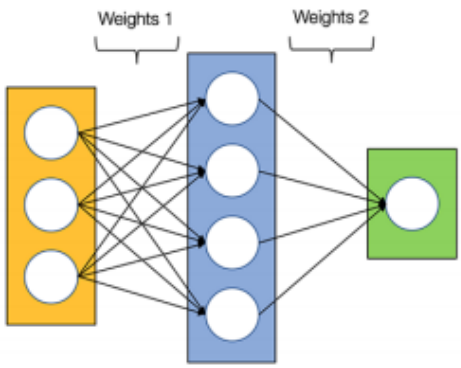
TU856 Year 3

Multi-layered Perceptron's (MLP) and Backpropagation

Problem 1:

For this problem we had to consider a MLP with a hidden layer of 4 sigmoid neurons and the following is the data set.

X1	X2	X3	Y
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0



Inputs and Outputs, I Used:

```
inputs = np.array([[0,0,1], [0,1,1], [1,0,1], [1,1,1]])
out = np.array([0,1,1,0])
```

Generalized inputs:

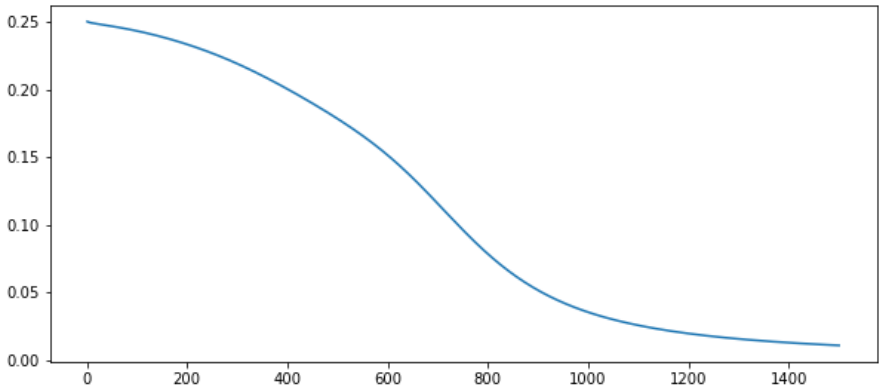
N = Number of neurons in the hidden layer

M = Number of Inputs

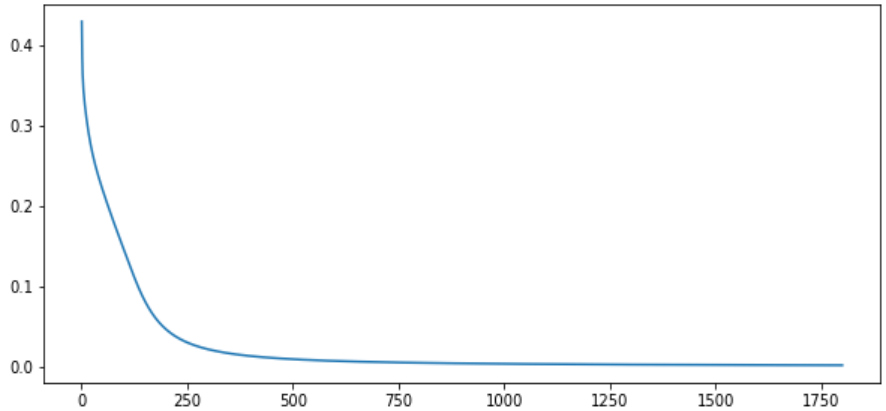
O = Number of outputs

```
N = 4
M = 3
O = 1
```

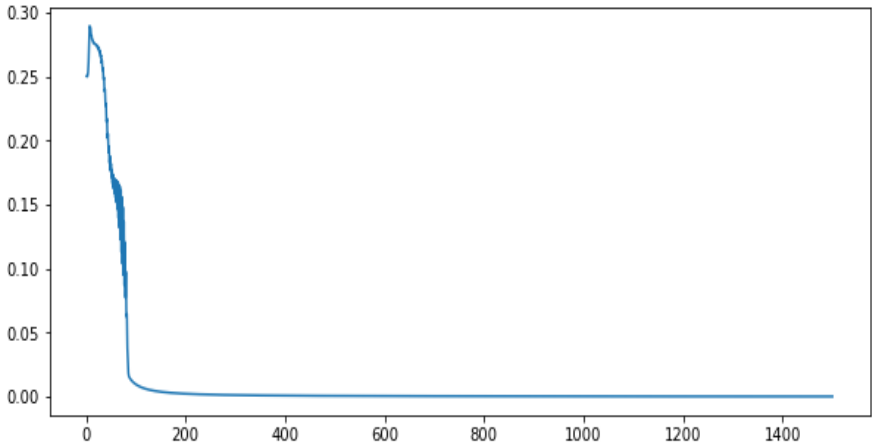
Result with a low learning rate (1.0):



Result with a medium learning rate (3.0):



Result with a high learning rate (20.0):



As you can see the higher the learning rate the more unstable it gets.

Problem 2 - Neural Network with 3 input and 2 output:

For this question I had to Try a MLP on the following training data. Vary the number of neurons in the hidden layer, experiment with learning rates and epochs. Analyse the training.

```
X_training=[[ 1,  1,  0],
            [ 1, -1, -1],
            [-1,  1,  1],
            [-1, -1,  1],
            [ 0,  1, -1],
            [ 0, -1, -1],
            [ 1,  1,  1]
           ]
y_training=[[1, 0],
            [0, 1],
            [1, 1],
            [1, 0],
            [1, 0],
            [1, 1],
            [1, 1]
           ]
```

Inputs and Outputs, I Used:

```
inputs = np.array([[1,1,0], [1,-1,-1], [-1,1,1], [-1,-1,1], [0,1,-1], [0,-1,-1], [1,1,1]])
out = np.array([[1,0],[0,1],[1,1],[1,0],[1,0],[1,1],[1,1]])
```

Generalized inputs:

N = Number of neurons in the hidden layer

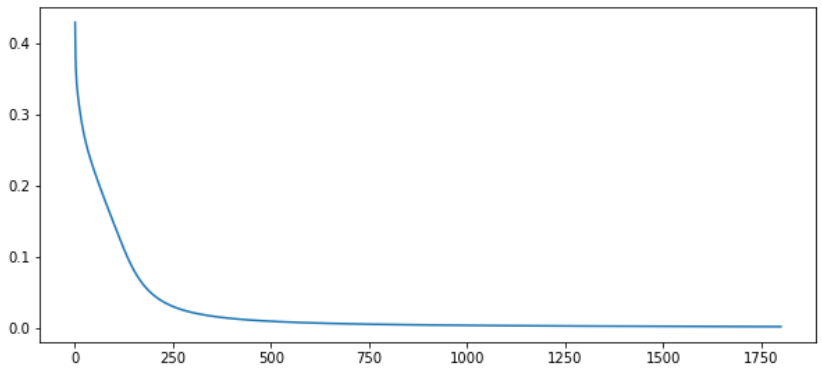
M = Number of Inputs

O = Number of outputs

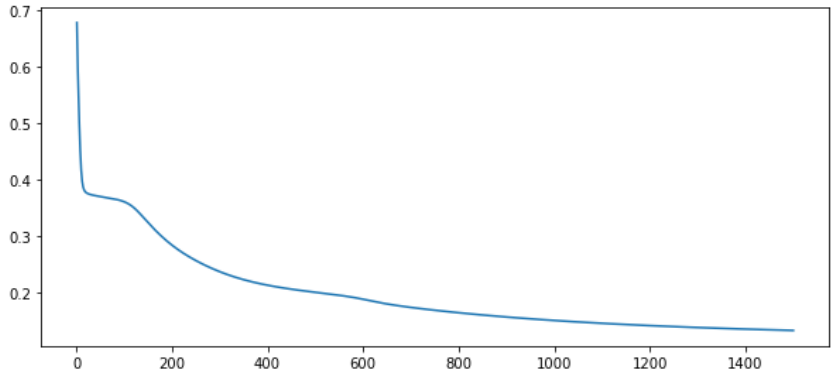
```
N = 4
M = 3
O = 2
```

For this problem I had to change the number of epochs, learning rate and the number of neurons in the hidden layer to see how it affects the outcome.

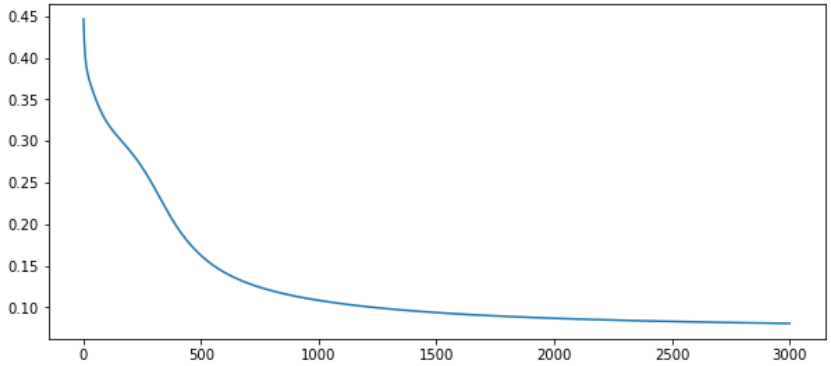
Result with 1800 epochs, 3.0 learning rate and 4 N:



Result with 1500 epochs, 2.0 learning rate and 1 N:



Result with 3000 epochs, 1.0 learning rate and 2 N:



As you can see from the outputs above when you start to change the number of epochs, the learning rate, and the number of neurons in the hidden layer your outcomes change noticeably. The change can most notably be seen when you change the number of neurons in the hidden layer.

Problem 3 - Transportation Mode Choice:

For this problem we were given this data:

Gender	Car ownership	Travel Cost	Income Level	Transportation mode
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car

The training data is supposed to be part of a transportation study regarding the mode choice to select bus, car or train among commuters along a major route in a city, gathered through a questionnaire study. For simplicity and clarity, we selected only 4 attributes. Attribute 'gender' is a binary type, while 'car ownership' is a quantitative integer. 'Travel cost/km' is a quantitative of ratio type but here it was converted into an ordinal type. 'Income level' is also an ordinal type. Train a neural network to predict the transport mode of a person, given the four attributes: gender, car ownership, travel cost, and income level.

Inputs and Outputs, I Used:

```
inputs = np.array([[1,0,0,0], [1,1,0,1], [0,1,0,1], [0,0,0,0],
                   [1,1,0,2], [1,0,2,1], [0,1,1,1], [0,1,2,2],
                   [1,2,2,1], [0,2,2,2], [0,0,2,1]])

out = np.array([[0,0,1],[0,0,1],[0,1,0],[0,0,1],[0,0,1],[0,1,0],[0,1,0],[1,0,0],[1,0,0],[1,0,0]])
```

Generalized inputs:

N = Number of neurons in the hidden layer

M = Number of Inputs

O = Number of outputs

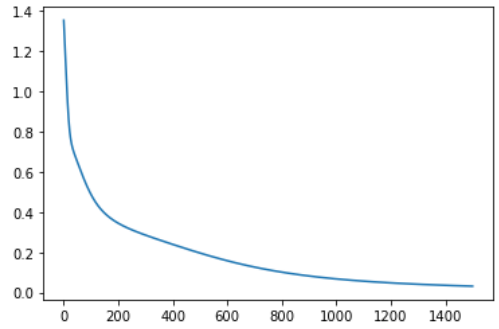
```
N = 4
M = 4
O = 3
```

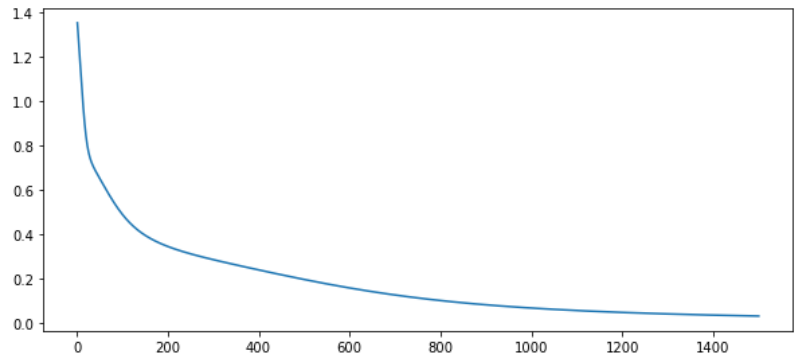
Result when I try to predict the Transportation Mode Choice of the following instance of data:

Female without car ownership, willing to pay expensive travel cost and having medium income level. (0,0,2,1).

```
Result for Female without car ownership, willing to pay expensive travel cost and having medium income level

0.1345812888843036  Bus
0.8791293783432251   Train
0.05275481466004327  Car
```





As you can see from the outputted result the mode of transport it predicts is 'Train'.

I also had to export the feedforward as a csv file. This is the code I used:

```
result = xor.feedforward(xs)
```

```
#Exporting the result of feedforward
df = pd.DataFrame(result)
df.to_csv('transport.csv', sep=',', encoding='utf-8', index=False)
```

This is a screenshot of the transport.csv file:

	A	B	C	D	E	F	G	H	I	J	K
1	0	1	2	3	4	5	6	7	8	9	10
2	0.001244	0.003978	0.068325	0.004924	0.014471	0.118088	0.216285	0.840859	0.877735	0.917415	0.150388
3	0.028597	0.089093	0.793679	0.132937	0.105903	0.817302	0.795943	0.169097	0.131303	0.087585	0.862154
4	0.988303	0.948967	0.184409	0.906996	0.862703	0.087026	0.05825	0.021951	0.021691	0.01785	0.058312

Problem 4 - Iris flower classification:

For this problem I had to read in data set iris.csv into a data frame using the panda's library. I then had to convert that data frame into a NumPy array in order to make it work with the XOR code I have.

I populated my inputs and outputs by reading the csv file into a data frame and turning that into a NumPy array:

```
#Problem 4
iris_df1 = pd.read_csv('iris_data.csv', header = None, usecols=[0,1,2,3])
inputs = iris_df1.to_numpy()

iris_df2 = pd.read_csv('iris_data.csv', header = None, usecols=[4,5,6])
out = iris_df2.to_numpy()
```

Generalized inputs:

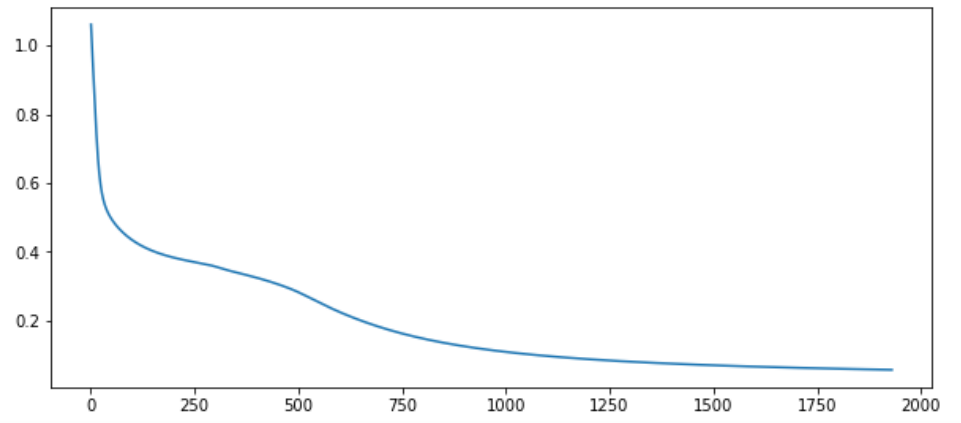
N = Number of neurons in the hidden layer

M = Number of Inputs

O = Number of outputs

```
N = 4
M = 4
O = 3
```

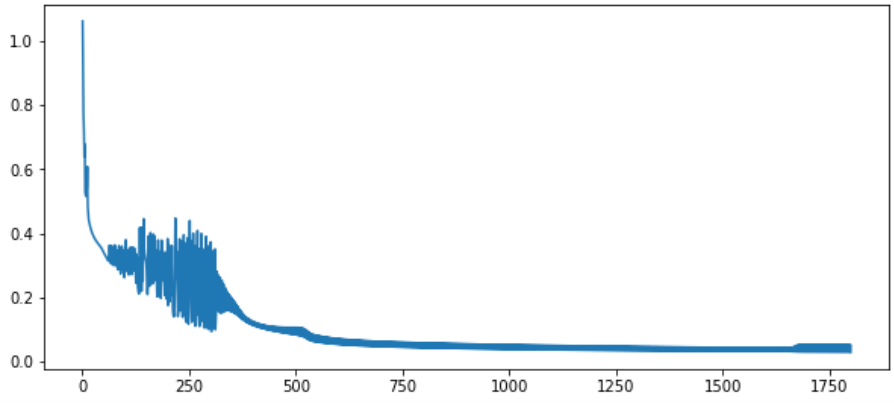
Result using 1930 epochs, 0.5 learning rate and 4 N:



The reason I used 1930 epochs, 0.5 learning rate and 4 N is because I found it gives the most stable result. I first had the learning rate higher but the graph showed it was very unstable:

Result using 1800 epochs, 4.0 learning rate and 4 N (Unstable output):

This graph has a high learning rate and as you can see the outcome is very unstable.



Conclusion:

After completing this assignment, I feel like my knowledge of multi-layered perceptron's and AI in general has improved greatly. At first, I found the assignment a bit challenging but the more I got through it, the better I understood it and the easier it became.