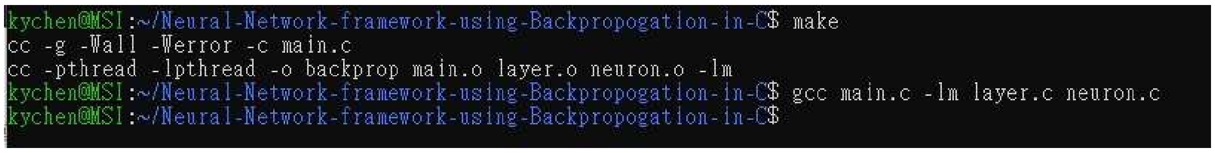
Assignment #1

通訊二陳冠羽110503524

(注1:這個補交檔案是因為發現別人都是一整個資料夾匯出，而我因為技術問題是一個檔案一個檔案匯出，所以改動只有程式資料夾多加幾個檔案和多一點點無聊打的新註解而已

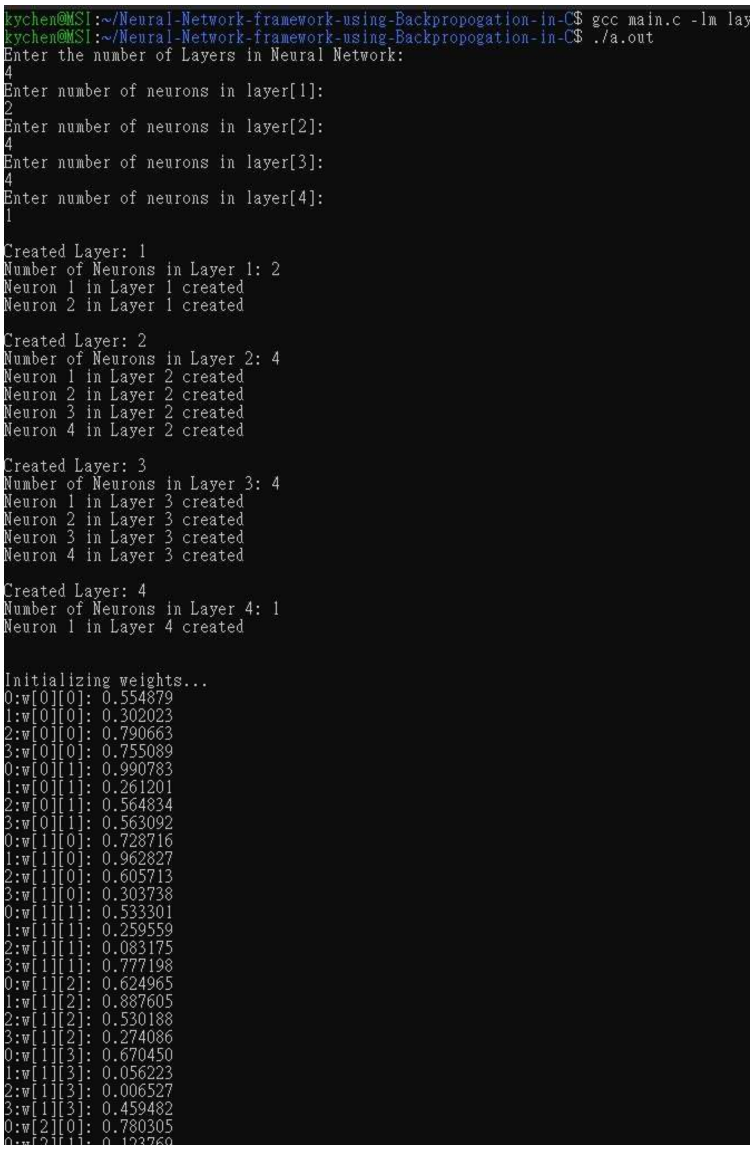
注2 : backpropaaa只是亂取名的執行檔，和a.out是同內容，無特別意義)

1. 編譯結果

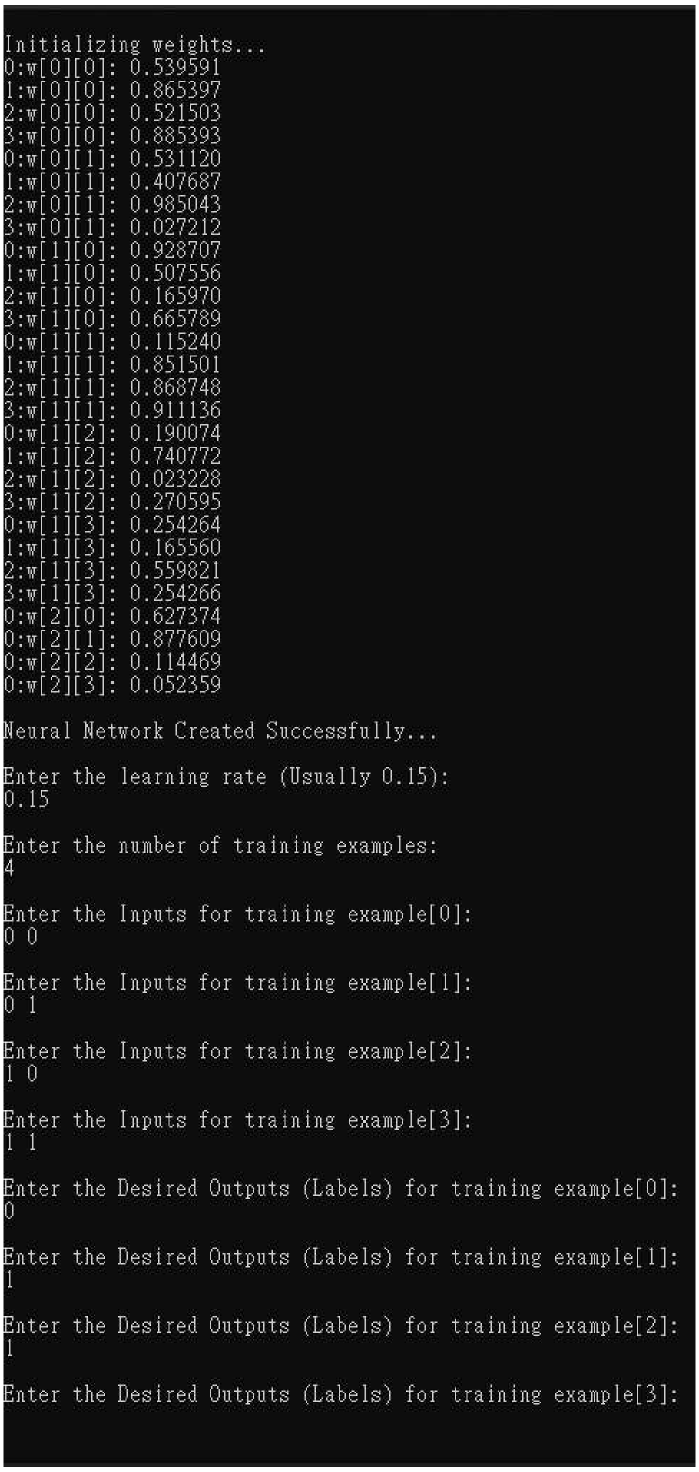


```
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$ make
cc -g -Wall -Werror -c main.c
cc -pthread -lpthread -o backprop main.o layer.o neuron.o -lm
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$ gcc main.c -lm layer.c neuron.c
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$
```

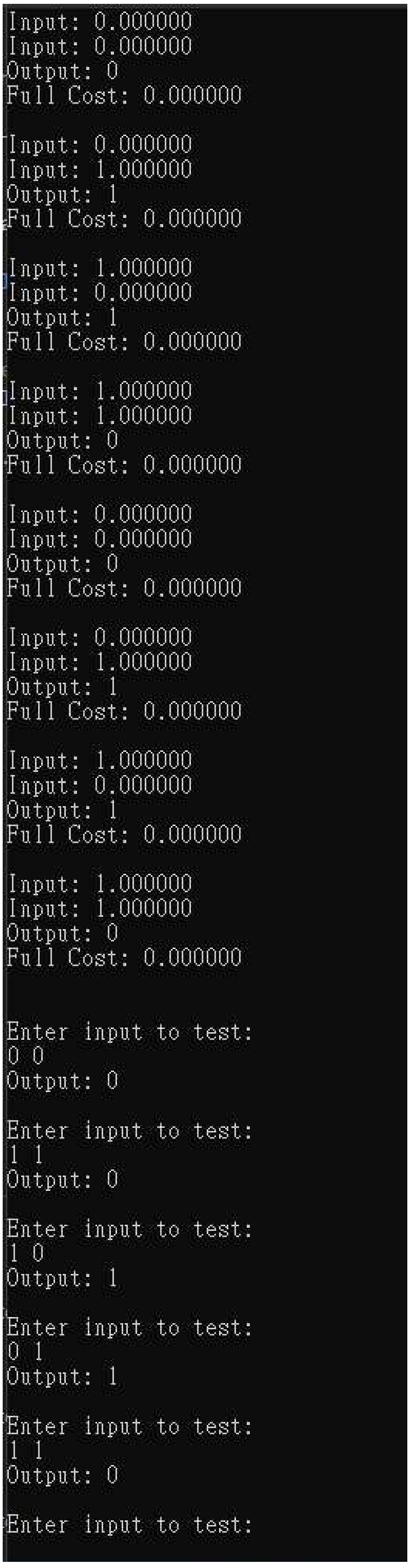
2. 執行結果



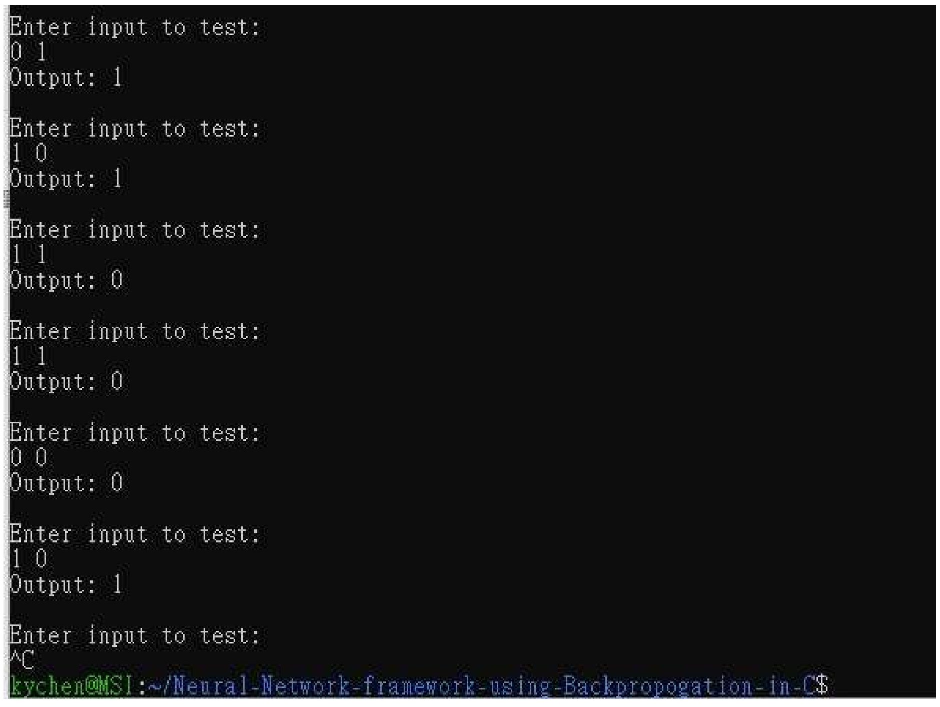
```
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$ gcc main.c -lm lay
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$ ./a.out
Enter the number of Layers in Neural Network:
4
Enter number of neurons in layer[1]:
2
Enter number of neurons in layer[2]:
4
Enter number of neurons in layer[3]:
4
Enter number of neurons in layer[4]:
1

Created Layer: 1
Number of Neurons in Layer 1: 2
Neuron 1 in Layer 1 created
Neuron 2 in Layer 1 created

Created Layer: 2
Number of Neurons in Layer 2: 4
Neuron 1 in Layer 2 created
Neuron 2 in Layer 2 created
Neuron 3 in Layer 2 created
Neuron 4 in Layer 2 created

Created Layer: 3
Number of Neurons in Layer 3: 4
Neuron 1 in Layer 3 created
Neuron 2 in Layer 3 created
Neuron 3 in Layer 3 created
Neuron 4 in Layer 3 created

Created Layer: 4
Number of Neurons in Layer 4: 1
Neuron 1 in Layer 4 created


Initializing weights...
0:w[0][0]: 0.554879
1:w[0][0]: 0.302023
2:w[0][0]: 0.790663
3:w[0][0]: 0.755089
0:w[0][1]: 0.990783
1:w[0][1]: 0.261201
2:w[0][1]: 0.564834
3:w[0][1]: 0.563092
0:w[1][0]: 0.728716
1:w[1][0]: 0.962827
2:w[1][0]: 0.605713
3:w[1][0]: 0.303738
0:w[1][1]: 0.533301
1:w[1][1]: 0.259559
2:w[1][1]: 0.083175
3:w[1][1]: 0.777198
0:w[1][2]: 0.624965
1:w[1][2]: 0.887605
2:w[1][2]: 0.530188
3:w[1][2]: 0.274086
0:w[1][3]: 0.670450
1:w[1][3]: 0.056223
2:w[1][3]: 0.006527
3:w[1][3]: 0.459482
0:w[2][0]: 0.780305
0:w[2][1]: 0.123769
```



```
Initializing weights...
0:w[0][0]: 0.539591
1:w[0][0]: 0.865397
2:w[0][0]: 0.521503
3:w[0][0]: 0.885393
0:w[0][1]: 0.531120
1:w[0][1]: 0.407687
2:w[0][1]: 0.985043
3:w[0][1]: 0.027212
0:w[1][0]: 0.928707
1:w[1][0]: 0.507556
2:w[1][0]: 0.165970
3:w[1][0]: 0.665789
0:w[1][1]: 0.115240
1:w[1][1]: 0.851501
2:w[1][1]: 0.868748
3:w[1][1]: 0.911136
0:w[1][2]: 0.190074
1:w[1][2]: 0.740772
2:w[1][2]: 0.023228
3:w[1][2]: 0.270595
0:w[1][3]: 0.254264
1:w[1][3]: 0.165560
2:w[1][3]: 0.559821
3:w[1][3]: 0.254266
0:w[2][0]: 0.627374
0:w[2][1]: 0.877609
0:w[2][2]: 0.114469
0:w[2][3]: 0.052359

Neural Network Created Successfully...

Enter the learning rate (Usually 0.15):
0.15

Enter the number of training examples:
4

Enter the Inputs for training example[0]:
0 0

Enter the Inputs for training example[1]:
0 1

Enter the Inputs for training example[2]:
1 0

Enter the Inputs for training example[3]:
1 1

Enter the Desired Outputs (Labels) for training example[0]:
0

Enter the Desired Outputs (Labels) for training example[1]:
1

Enter the Desired Outputs (Labels) for training example[2]:
1

Enter the Desired Outputs (Labels) for training example[3]:
```



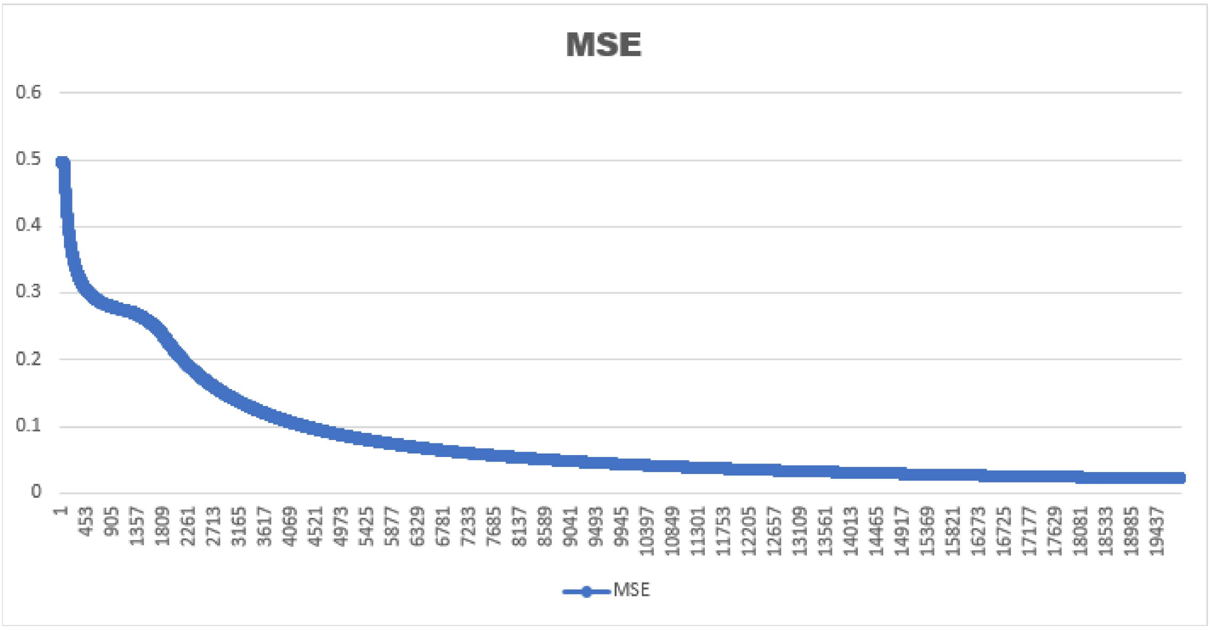
```
Input: 0.000000
Input: 0.000000
Output: 0
Full Cost: 0.000000

Input: 0.000000
Input: 1.000000
Output: 1
Full Cost: 0.000000

Input: 1.000000
Input: 0.000000
Output: 1
Full Cost: 0.000000

Input: 1.000000
Input: 1.000000
Output: 0
Full Cost: 0.000000

Input: 0.000000
Input: 0.000000
Output: 0
Full Cost: 0.000000

Input: 0.000000
Input: 1.000000
Output: 1
Full Cost: 0.000000

Input: 1.000000
Input: 0.000000
Output: 1
Full Cost: 0.000000

Input: 1.000000
Input: 1.000000
Output: 0
Full Cost: 0.000000


Enter input to test:
0 0
Output: 0

Enter input to test:
1 1
Output: 0

Enter input to test:
1 0
Output: 1

Enter input to test:
0 1
Output: 1

Enter input to test:
1 1
Output: 0

Enter input to test:
```



```
Enter input to test:
0 1
Output: 1

Enter input to test:
1 0
Output: 1

Enter input to test:
1 1
Output: 0

Enter input to test:
1 1
Output: 0

Enter input to test:
0 0
Output: 0

Enter input to test:
1 0
Output: 1

Enter input to test:
^C
kychen@MSI:~/Neural-Network-framework-using-Backpropogation-in-C$
```

3. 分析

可以看到最後執行結果電腦有成功學習，輸入對應的 00 01 11 10有得到 0 1 0 1。
我以機器學習時的錯誤(cost)以mean square error作圖得到下圖，可見學完了錯誤值有收斂。
而中間的凸起是照公式調整權重時意外歪掉，但可以看到再過陣子又收斂回來了，這是因機器學習只是照公式不停條權重，並不是經過人的判斷，所以一剛好數字對了跑掉也很正常。



4. 遇到問題與解決

一開始我ubuntu開不起來，之後查了很多資料其中像是參考資料1等等才發現是windows的虛擬機器平台要打開。

再來是各種新的語法要適應。git clone後我不知道要如何修改code，我所查到的就是用vi去改，但這介面實在是不方便。和另一位也不代會的同學搞了一陣子，看了參考資料2~5等等才知道要用code .和各種要下載的配件和編譯完如果沒設定執行檔名稱預設會是a.out。

要分析數據時也有資料要如何匯出的問題。一開始我想用手打，但資料太多了我會瘋掉，之後改轉txt，但想到一半發現可以直接用csv檔開，就可以變excel直接做圖了。

5. 參考資料

1.https://www.jianshu.com/p/be76a0f08dc7
2.https://blog.csdn.net/chenxi_li/article/details/93913210
3.https://code.visualstudio.com/docs/remote/wsl
4.https://learn.microsoft.com/zh-tw/windows/wsl/tutorials/wsl-vscode?fbclid=IwAR1xJMQ71avuA6hkQowFM79KnmgzO_99q2-5OceXsPtXi6Q-kIi_HwbW0Mg
5.https://blog.jaycetyle.com/2015/01/linux-gcc-makefile/?fbclid=IwAR3OSF6S-yTc55OCIraIQE31zBvPc22PL3CtWOxAtcHbqDnB1OdHdCPyT-E
6.老師的ppt
7.https://github.com/mayurbhole/Neural-Network-framework-using-Backpropogation-in-C

在此感謝參考資料7的code!!