110503527 通訊二陳廷宇

1. 編譯結果

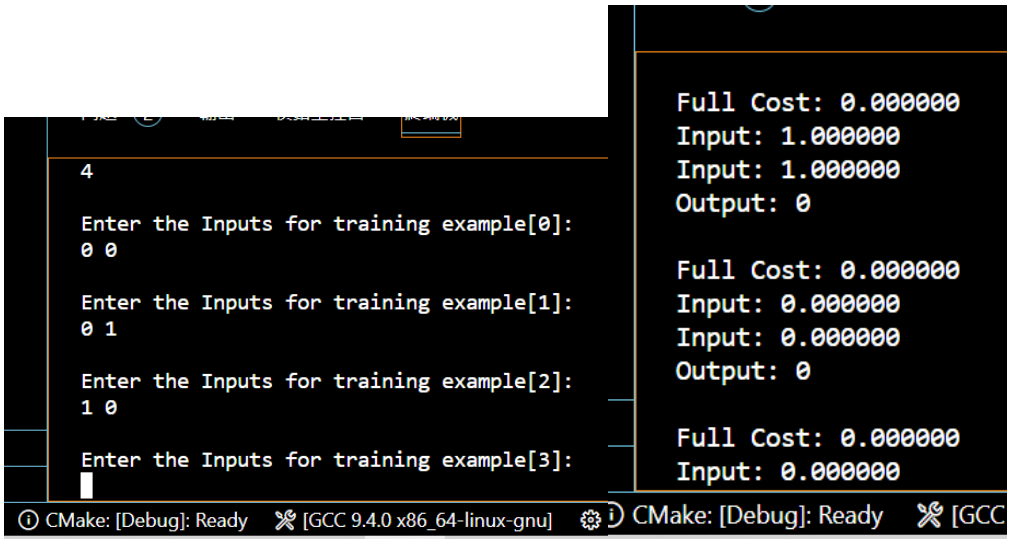
```
tingyu@LAPTOP-3KGQATTK:~$ cd /home/tingyu/Neural-Network-framework-using-Backpropogation-in-C/
tingyu@LAPTOP-3KGQATTK:~/Neural-Network-framework-using-Backpropogation-in-C$ make
cc -g -Wall -Werror -c main.c
cc -pthread -lpthread -o backprop main.o layer.o neuron.o -lm
tingyu@LAPTOP-3KGQATTK:~/Neural-Network-framework-using-Backpropogation-in-C$ gcc main.c -lm layer.c neuron.c
tingyu@LAPTOP-3KGQATTK:~/Neural-Network-framework-using-Backpropogation-in-C$
```

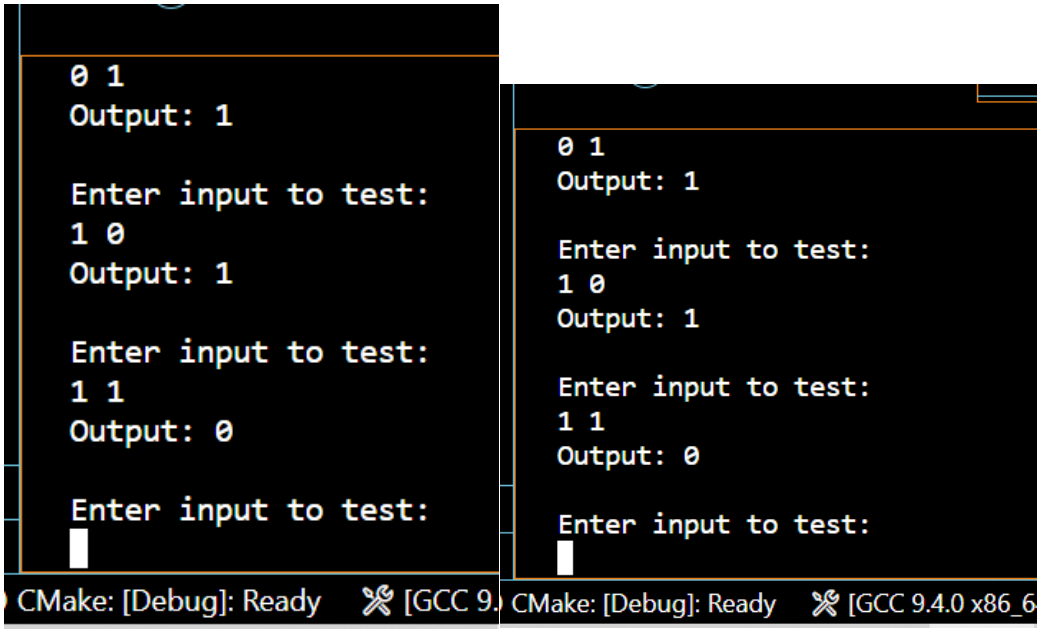
上圖為執行 make 與 gcc 編譯後之結果，編譯結果並無異狀。

2. 執行結果

```
tingyu@LAPTOP-3KGQATTK:~/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
```

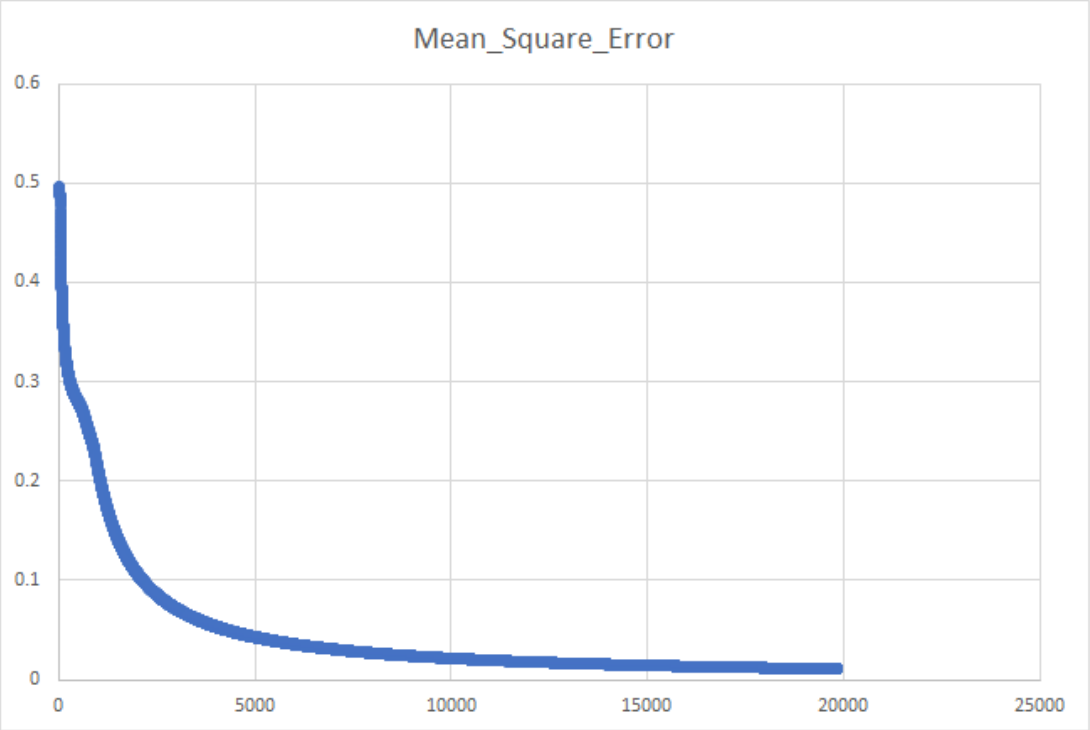
```
3:w[1][3]: 0.955277
0:w[2][0]: 0.089118
0:w[2][1]: 0.142314
0:w[2][2]: 0.304531
0:w[2][3]: 0.448729

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]:
CMake: [Debug]: Ready
[GCC 9.4.0 x86_64-linux-gnu]
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
CMake: [Debug]: Ready
[GCC


0 1
Output: 1
Enter input to test:
1 0
Output: 1
Enter input to test:
1 1
Output: 0
Enter input to test:
CMake: [Debug]: Ready
[GCC 9.
0 1
Output: 1
Enter input to test:
1 0
Output: 1
Enter input to test:
1 1
Output: 0
Enter input to test:
CMake: [Debug]: Ready
[GCC 9.4.0 x86_6

以上六張圖片為編譯完成後執行之結果，可以看到與預期一樣程式會先要求使用者輸入欲學習的資料量與資料，並依據使用者所輸入的資料進行學習，本次的 code 採用 20000 次的迴圈。成功學習後，程式會提示使用者輸入測試用的數據，檢測是否完成學習。

3. 誤差分析



此圖為 NN 學習當中所產生的誤差值之分析，本次分析採用均方誤差分析(Mean Square Error)公式進行。其公式如下

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

透過程式內的 Compute_MSE 之函式，將每筆誤差值帶入公式定求出每次學習的均方誤差再寫入 data.csv 中紀錄，最後由 excel 進行作圖。圖中我們可以發現隨著迴圈次數增加，終端機所學習的誤差逐漸縮小、收斂，最後趨向正確的數值。

4. 未來展望

- 此次為第一階段之作業，在第二階段會將程式內以陣列宣告之部分用指位器替代。
- 在後續的第二次驗收時希望能夠嘗試多 Bits 的運算而不只侷限於 2-Bits。
- 後續將嘗試優化程式提高其運作之效率

5. 假設/提問

這邊想提出一點問題，根據我的推測，如果增加迴圈運行的圈數，能否讓結果更加收斂至欲得之數據，那請問能否真的讓數據收斂至完全正確之值?