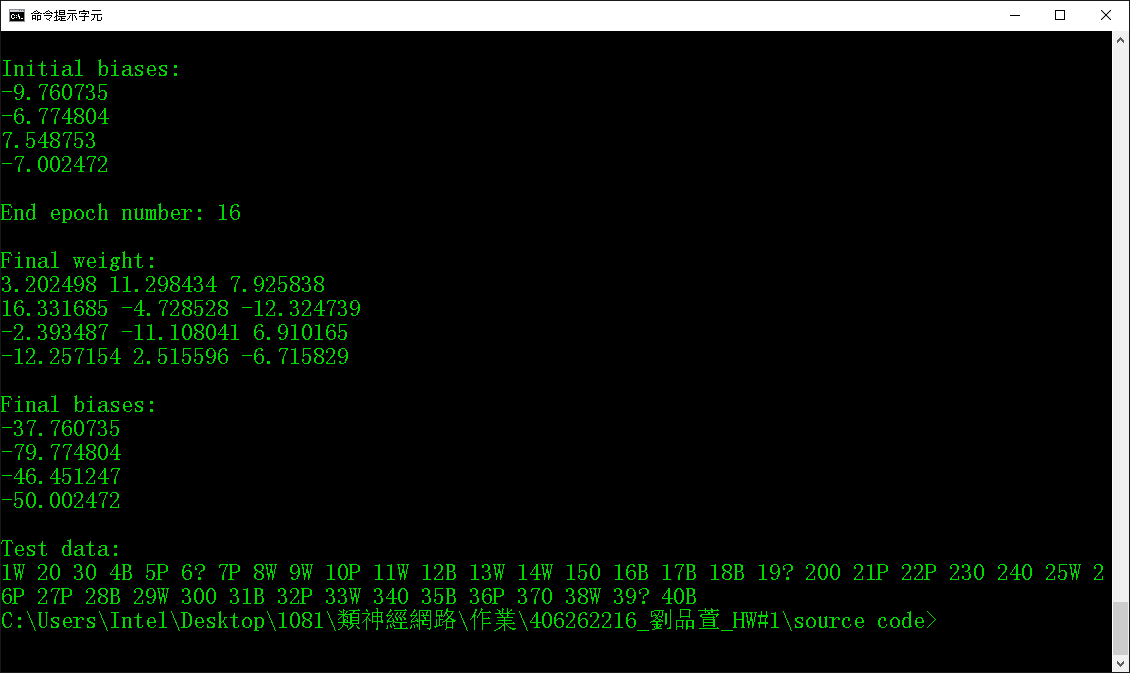
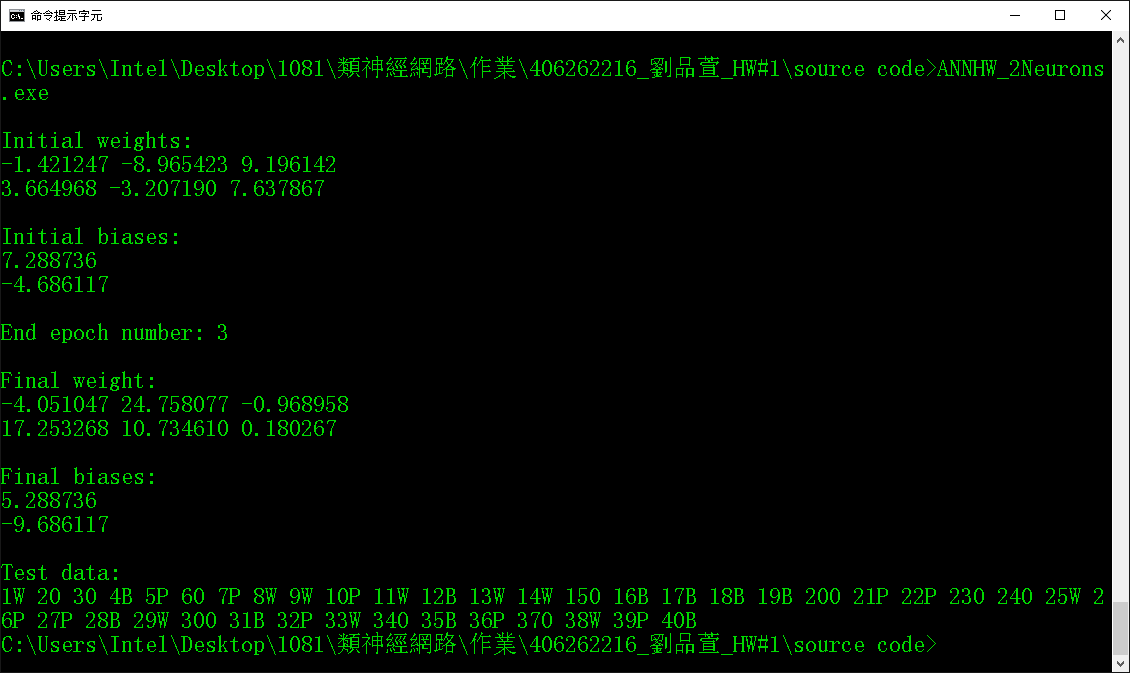
**類神經網路 程式作業一 報告**

1. **Compare the performance of your two-neuron perceptron and four-neuron perceptron.**

因為此訓練完成的Perceptron Network，在測試Test data的時候，只會比對他的輸出a是否有符合已經設定好的種類的target值，有的話才會被歸類成某類，沒有的會就會是 ?，此現象在圖(1)的四個神經元的情況中更為明顯：



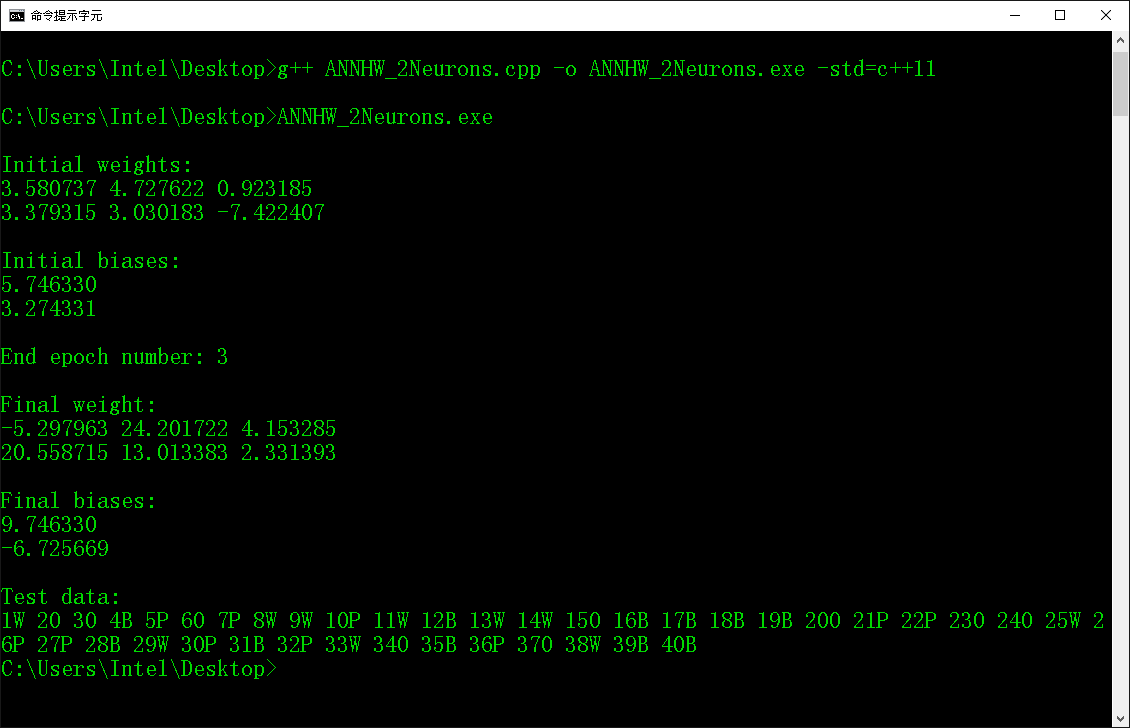
圖(1) 4個神經元的情況下，第6個、第19個、第39個Test data無法分類到某類別



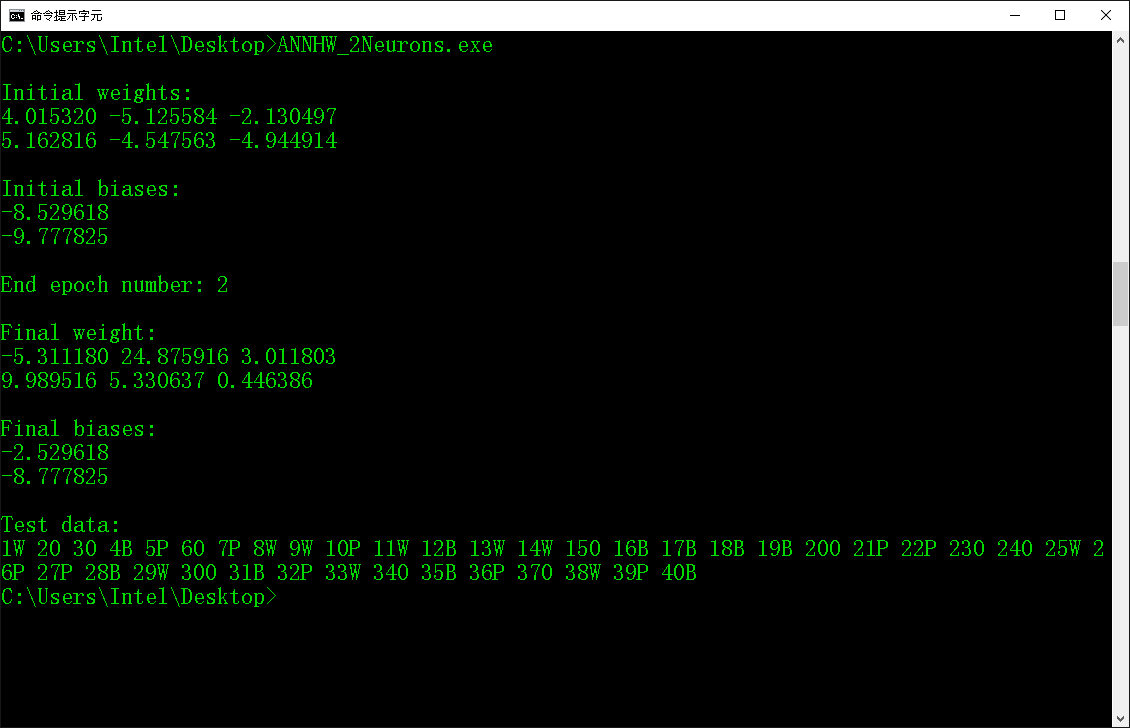
圖(2) 兩個神經元的情況下，40個Test data都有類別

1. **Compare the performance of different initial weights/biases.**

利用自定義的函數double fRand(double fMin, double fMax)，產生某隨機M及b（M及b的內容值控制在10~-10之間）。使得每一次的執行，M 和 b 在訓練時發生的更新變化不同，進而使得訓練Perceptron Network所花費的世代數不同。帶入Test data後產生的結果，也造成有時候會發生小幅度的差異（差不多1至2個）。如下圖(3)、圖(4)所示：



圖(3) 花費3個世代訓練完畢



圖(4) 花費2個世代訓練完畢

1. **Compare the performance of different learning rates.**

我針對四個神經元的Perceptron Network，在cpp檔內define了一個名為learningRate的變數，並加上迴圈，使得在某一個特定learning rate的情況下執行了10000次的訓練（每一次所使用的W及b都為亂數產生）。

其learning rate包括了1(LearningRate1In4Neurons.txt)、0.75(LearningRate1In4Neurons.txt)、0.5(LearningRate1In4Neurons.txt)、0.25(LearningRate1In4Neurons.txt)、0.01(LearningRate1In4Neurons.txt)、0.001(LearningRate1In4Neurons.txt)六種。

而在訓練完這六種不同learning rate 10000次的情況下，產生的差異如下：

在learning rates 為 1 的狀況時：平均epoch數為8

在learning rates 為 0.75 的狀況時：平均epoch數為10

在learning rates 為 0.5 的狀況時：平均epoch數為11

在learning rates 為 0.25 的狀況時：平均epoch數為10

在learning rates 為 0.01 的狀況時：平均epoch數為29

在learning rates 為 0.001的狀況時：平均epoch數為298

從以上數據分析可得知，當rate從1在逐一情況降低至0.25的時候，平均花費的epoch有往上升的趨勢，但是仍有一些小浮動。而在分析的結果中，可以看出最明顯的差異是在learning rates設置為0.01與0.001時epoch數大幅度增加。因此可知learning rate與epoch數有一定關係，通常當learning rate降低的時候，epoch數會增加。

（上述分析的數據都以檔案寫出的方式寫入各自txt檔案中）