

Report

0616027 陳昱銘

1.這次作業和前面差不多，也是用 python 寫的，在終端機打
python back.py

會跑出很多個 weight，分別是 1-4 層的 weight，最一開始的
weight 是隨機選擇的，經過 10000 次之後，weight 會被更新，
之後還有 20000, 30000,...次，可以看到 loss 逐漸變小

```
epochs 10000 loss: 1.991648e-05  
Accuracy : 100 %  
epochs 20000 loss: 4.669134e-06  
Accuracy : 100 %  
epochs 30000 loss: 2.963493e-06  
Accuracy : 100 %  
epochs 40000 loss: 2.226889e-06  
Accuracy : 100 %  
epochs 50000 loss: 1.784686e-06  
Accuracy : 100 %
```

Weight 就不印出來了，太多

2.

a. forward 在 think function 裡，最後也是由這裡和 weight 分
類出 label

b. sigmoid

def sigmoid(self, x):

return 1 / (1 + exp(-x))

c. back propagation

```
def sigmoid_derivative(self, x):
```

```
    return x * (1 - x)
```

這個接在 forward 之後，由第四層推回第一層，慢慢修正

weight，在第 30-40 行。

d. 藉由前一步的 back propagation，調整之前的 weight，在第 43-51 行。

e. 兩張圖分別在 predict.png 和 truth.png

f. loss 越來越小

```
[ 1.6924546 ]]  
epochs 10000 loss: 1.991648e-05  
epochs 20000 loss: 4.669134e-06  
epochs 30000 loss: 2.963493e-06  
epochs 40000 loss: 2.226889e-06  
epochs 50000 loss: 1.784686e-06  
epochs 60000 loss: 1.477619e-06  
epochs 70000 loss: 1.247952e-06  
epochs 80000 loss: 1.068827e-06  
epochs 90000 loss: 9.254196e-07  
epochs 100000 loss: 8.085148e-07
```

g. prediction

在 predict.txt 中，我有印出一個陣列代表從第 1-100 筆的預測結果，數值是 0.99xx 多的，代表預測結果接近 1，若是 0.00xx 多的，代表預測結果較接近 0