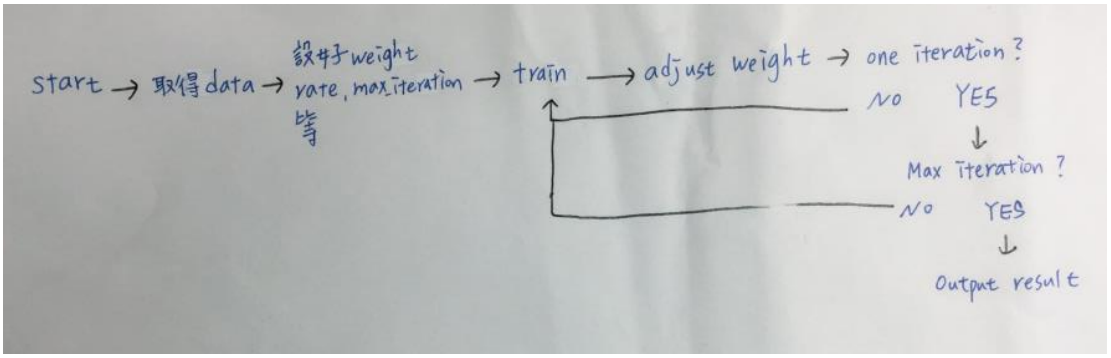
做法

這題我使用了一層 hidden layer ，250 個 node，開始 train 之後，便是接受 input，其中採用的 active function 是 sigmoid function(或者 reLu)，經過跟 weight 內積之後，output 的輸出跟正確答案比較後調整 weight，直到我指定的最大次數。

Flowchart:

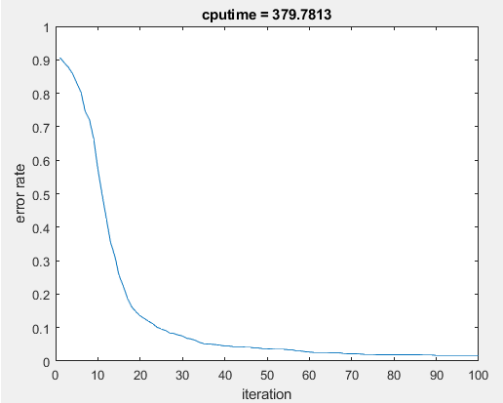


實驗(1)

Output:

a. (sigmoid function)

(1) Plot the figure of average error vs. iteration, CPU time in learning.



(2) &(3)

(4)confusion matrix (百分比)

	1	2	3	4	5	6	7	8	9	10
1	8.3000	0	0.1000	0.1000	0	0.1000	0.2000	0	0.1000	0
2	0	12.3000	0.2000	0	0.1000	0	0	0.1000	0	0
3	0	0	10.1000	0.3000	0	0.1000	0.1000	0.3000	0.1000	0
4	0	0.1000	0	7.6000	0	0	0	0.4000	0.2000	0.2000
5	0	0	0.1000	0.1000	9.3000	0	0	0.3000	0.4000	0.1000
6	0	0	0	1.6000	0.1000	7.5000	0.4000	0.1000	0.5000	0.3000
7	0.1000	0.1000	0.1000	0.2000	0.3000	0.2000	8	0	0.2000	0
8	0.1000	0	0.5000	0.1000	0	0.4000	0	8.4000	0.2000	0.8000
9	0	0.1000	0.4000	0.6000	0	0.3000	0	0	6.8000	0
10	0	0	0.1000	0.1000	1.2000	0.1000	0	0.3000	0.4000	8

討論:

How to determine the hidden node number ?

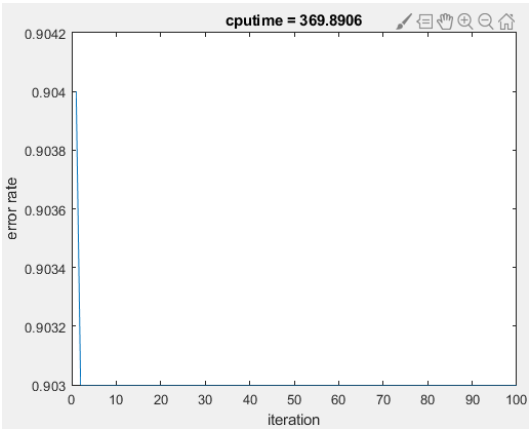
我使用了一層 hidden layer , 250 個 nodes, 會這樣使用是因為我之前用了兩層以上的 layer，發現效率很慢，而且準確率不高，似乎有 overfitting 的現象，後來發現這樣使用準確率還不錯，而且蠻快的。

Describe any phenomenon you watched

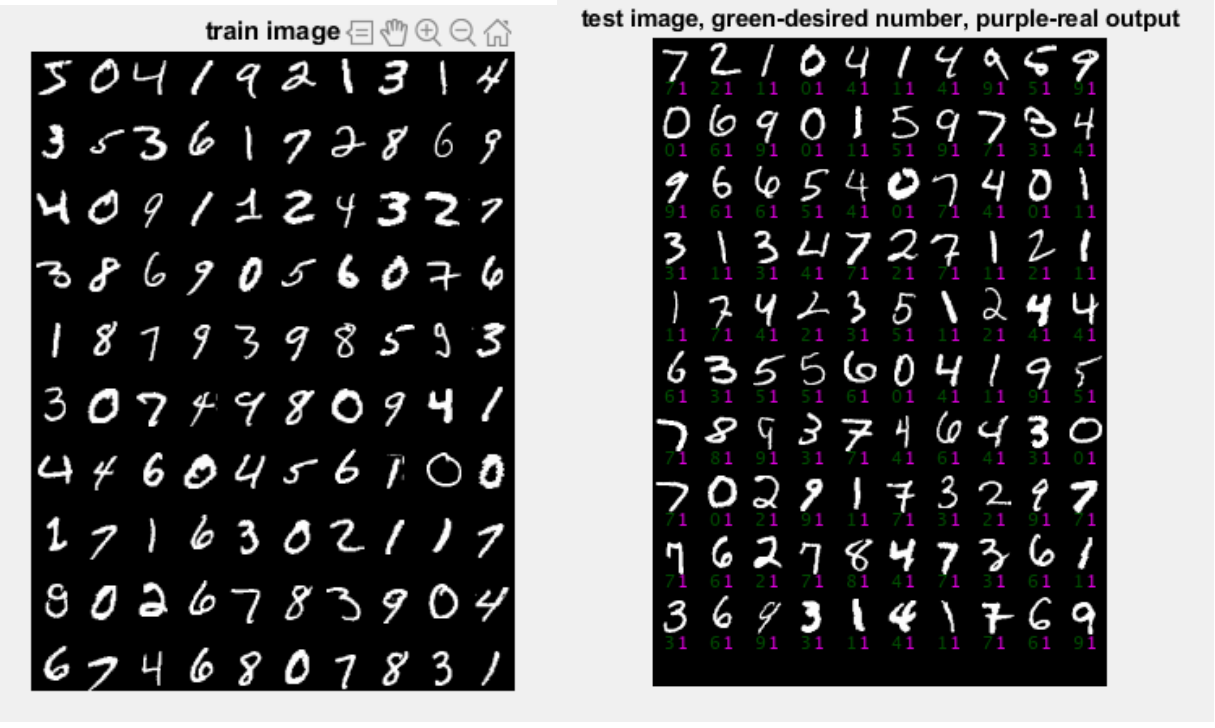
試過許多不同的 node 數量，發現 200-400 個 node 做出來的準確率試差不多的。

b. (relu function)

(1) Plot the figure of average error vs. iteration, CPU time in learning.



(2)&(3)



(4) confusion matrix (百分比)

	1	2	3	4	5	6	7	8	9	10
1	8.5000	12.6000	11.6000	10.7000	11	8.7000	8.7000	9.9000	8.9000	9.4000
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

討論:

How to determine the hidden node number ?

因為是跟 sigmoid 的比較，所以就跟之前用的一樣。

Describe any phenomenon you watched

試過許多不同的 node 數量或改一些 rate 的參數，發現結果都很差，如果要用 relu 應該要重新整個設計過，才能達到高的準確率。

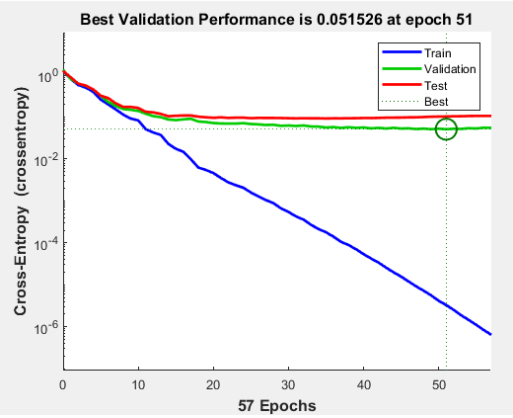
與 sigmoid 的收斂有什麼不一樣?

很快就達到收斂，但是效果很差。

c.(toolbox)

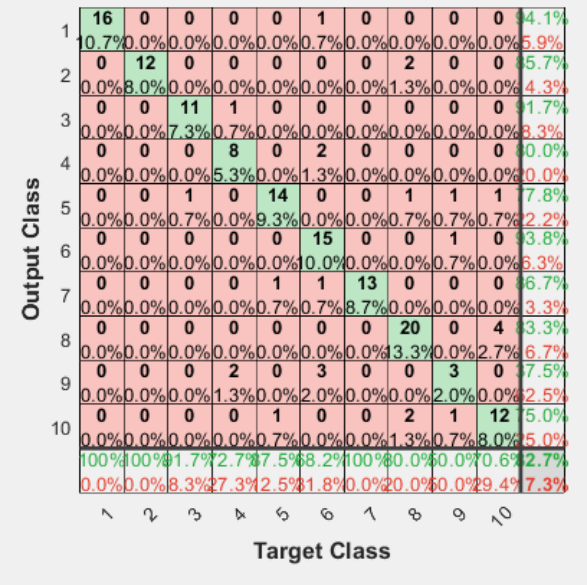
(1) Plot the figure of average error vs. iteration, CPU time in learning.

Cputime is on window



(2)&(3)

(4) confusion matrix



討論

How to determine the hidden node number ?

因為是跟之前的比較，所以就跟之前用的一樣。

Describe any phenomenon you watched

試過許多不同的 node 數量，發現 200-400 個 node 做出來的準確率試差不多的。

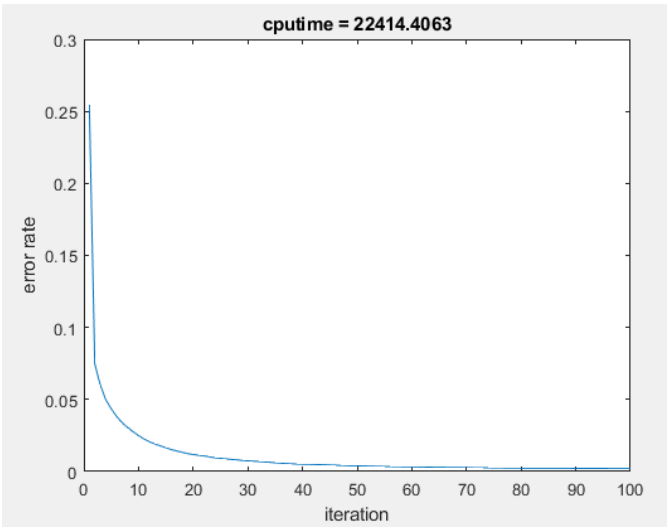
與不用 toolbox 有什麼不一樣?

除了只要把 input 跟環境設好非常方便外，toolbox 的效率很高，準確率也還不錯。

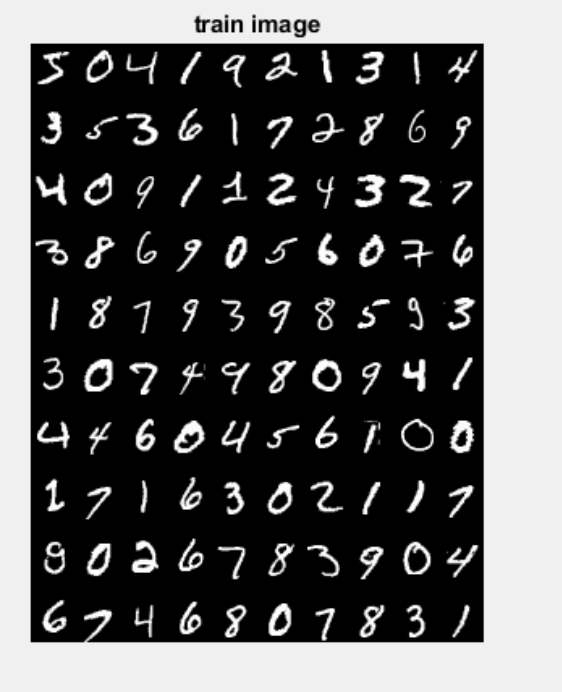
實驗(2)

a. (sigmoid function)

(1) Plot the figure of average error vs. iteration, CPU time in learning.



(2)&(3)



(4) confusion matrix (百分比)

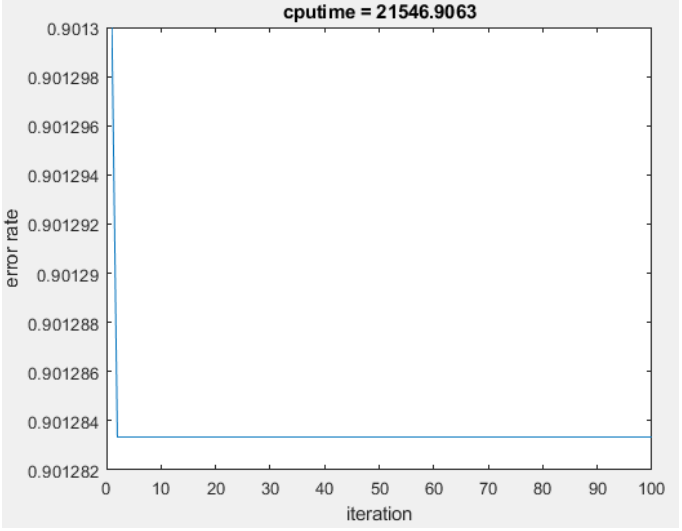
	1	2	3	4	5	6	7	8	9	10
1	9.7300	0	0.0500	0	0	0.0200	0.0500	0	0.0400	0.0300
2	0	11.2400	0.0100	0	0	0	0.0200	0.0400	0	0.0200
3	0.0100	0.0400	10.1000	0.0100	0.0200	0	0	0.0900	0.0400	0
4	0.0100	0	0	10.0100	0.0100	0.0900	0.0100	0.0200	0.0200	0.0500
5	0	0	0.0100	0	9.6600	0.0100	0.0100	0.0100	0.0400	0.0800
6	0.0100	0.0100	0	0.0300	0.0200	8.7700	0.0300	0	0.0400	0.0100
7	0.0100	0.0200	0	0.0100	0.0500	0.0100	9.4300	0	0.0100	0.0100
8	0.0100	0.0200	0.0700	0.0200	0	0	0	10.0400	0.0300	0.0400
9	0.0100	0.0200	0.0700	0.0200	0	0	0.0300	0.0200	9.4900	0.0100
10	0.0100	0	0.0100	0	0.0600	0.0200	0	0.0600	0.0300	9.8400
11										

討論:

除了資料的比數以外都和實驗一一樣，而準確率在實驗一中雖然已經很高了，但 train 的比數拉大，準確率還是上升了。

b. (reLU)

(1) Plot the figure of average error vs. iteration, CPU time in learning.



(2)&(3)

(4)confusion matrix

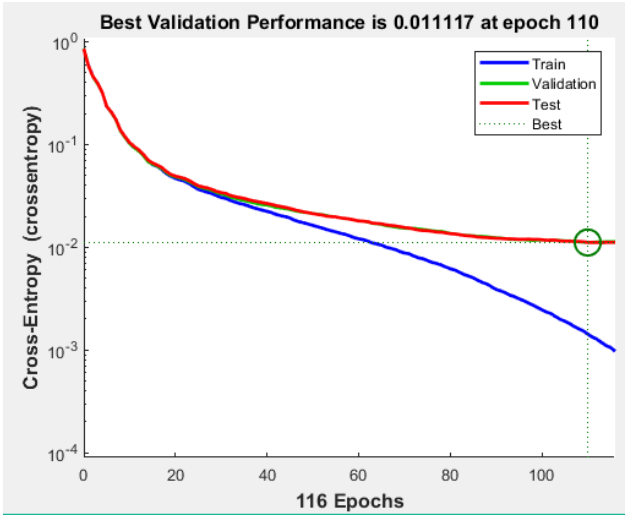
	1	2	3	4	5	6	7	8	9	10
1	9.8000	11.3500	10.3200	10.1000	9.8200	8.9200	9.5800	10.2800	9.7400	10.0900
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

討論:

除了資料的比數以外都和實驗一一樣，而準確率還是沒有上升，所以可以確定要用 relu 就要重新設計 model。

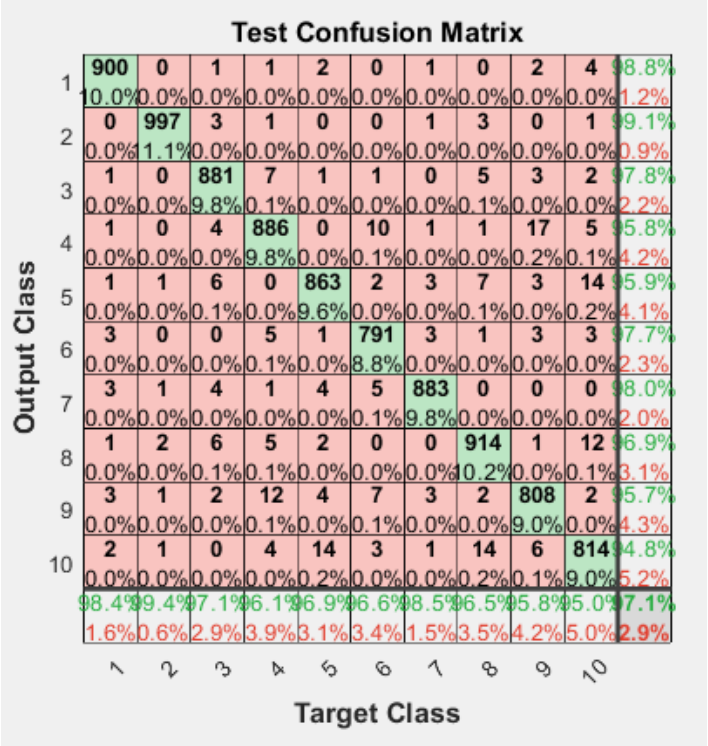
c.(toolbox)

(1) Plot the figure of average error vs. iteration, CPU time in learning.



(2)&(3)

(4) confusion matrix



討論:

基本上除了 train 的時間稍微拉長一點以外，還有準確率稍微上升，沒有什麼差太多的地方。