MLDS Homework 1 Report

學號：b05902031, 系級：資工二, 姓名：謝議霆, 學號：b05902008, 系級：資工二, 姓名：王行健

1-1-1 Simulate a Function

- 函數 : $y = \cos^2(x)(\sum_{i=1}^{100} \frac{2\sin(ix)}{(-1)^i i\pi} + \sum_{j=1}^{100} \frac{2\sin(\frac{jx}{2})}{(-1)^j \frac{j\pi}{2}})$
- DNN Model :

dense_1_input : InputLayer	input :	(None, 1)
	output :	(None, 1)
dense_2 : Dense	input :	(None, 1)
	output :	(None, 15)
dense_3 : Dense	input :	(None, 15)
	output :	(None, 30)
dense_4 : Dense	input :	(None, 30)
	output :	(None, 30)
dense_5 : Dense	input :	(None, 30)
	output :	(None, 15)
dense_6 : Dense	input :	(None, 15)
	output :	(None, 1)

Total = 1*16+15*31+30*31+30*16+15*2 = 1921

dense_1_input : InputLayer	input :	(None, 1)
	output :	(None, 1)
dense_2 : Dense	input :	(None, 1)
	output :	(None, 20)
dense_3 : Dense	input :	(None, 20)
	output :	(None, 45)
dense_4 : Dense	input :	(None, 45)
	output :	(None, 20)
dense_5 : Dense	input :	(None, 20)
	output :	(None, 1)

Total = 1*21+20*46+45*21+20*2 = 1926

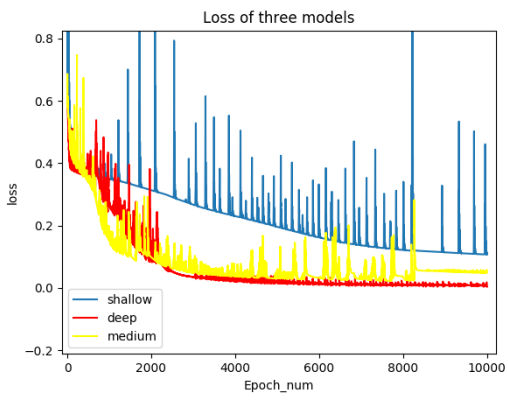
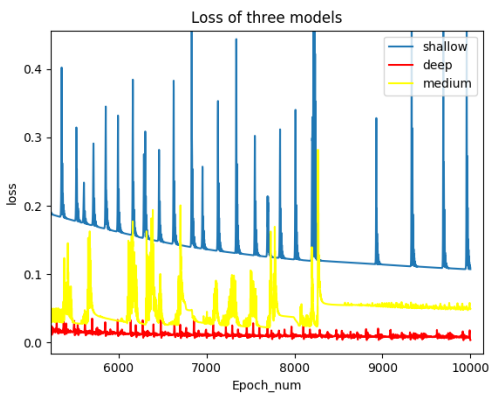
dense_1_input : InputLayer	input :	(None, 1)
	output :	(None, 1)
dense_2 : Dense	input :	(None, 1)
	output :	(None, 640)
dense_3 : Dense	input :	(None, 640)
	output :	(None, 1)

Total = 1*641+640*2 = 1921

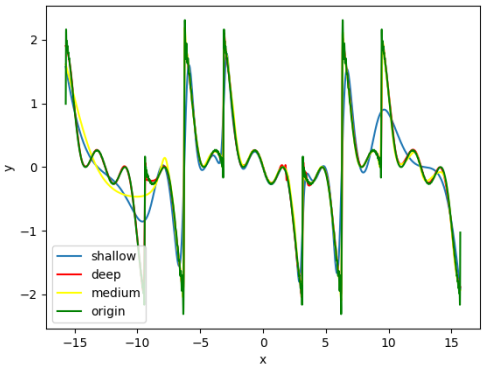
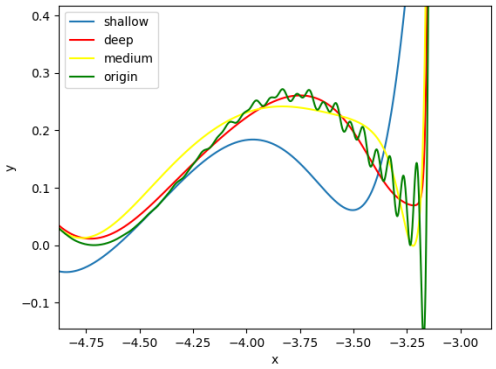
從左到右命名三個model為 **deep**, **medium**, 和 **shallow** ，各有 **1921**, **1926**, 和 **1921** 個參數.

Activation function : **tanh**

三個model的loss

Loss	Zoom in
	

三個model的預測

Predict	Zoom in
	

Comments

- Loss : 我們可以透過圖觀察出，deep和median的模型都比shallow的早收斂，loss也降得比shallow的低．最後比較deep和median的模型，雖然一開始還很難看出來誰的loss比較低，但是在收斂過後可以看出deep的loss比median的低，結論是越深的模型在這個函數上的loss越低．
- Prediction : 我們可以先看到shallow預測這個函數的結果不太好，接著比較deep和median的模型，**大部分**的函數區段是deep做得比較好，如右圖的Zoom in，少部分是median做得比較好，但是總的來說，越深的模型能predict出越好的結果．

1-1-2 Train on Actual Tasks —— MNIST

CNN model

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	1
	out_channel :	4

Maxpool2d(2,2)

Flatten

dense_1 : Dense	input :	(None, 784)
	output :	(None, 11)

dense_2 : Dense	input :	(None, 11)
	output :	(None, 10)

Total = 8859

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	1
	out_channel :	4

Maxpool2d(2,2)

Maxpool2d(2,2)

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	4
	out_channel :	8

Flatten

dense_1 : Dense	input :	(None, 392)
	output :	(None, 19)

dense_2 : Dense	input :	(None, 19)
	output :	(None, 10)

Total = 8579

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	1
	out_channel :	4

Maxpool2d(2,2)

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	4
	out_channel :	8

Maxpool2d(2,2)

Convolution_1_input : InputLayer Kernel_size = 5 Padding = 2	in_channel :	8
	out_channel :	12

Maxpool2d(2,2)

Flatten

dense_1 : Dense	input :	(None, 588)
	output :	(None, 9)

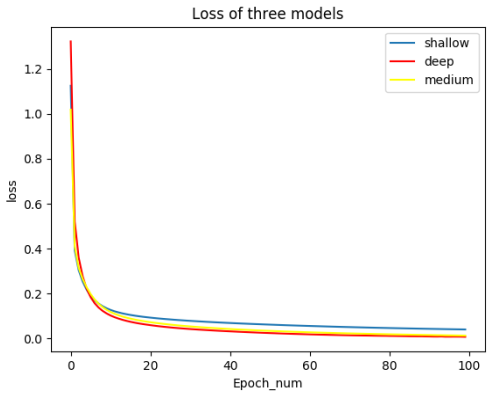
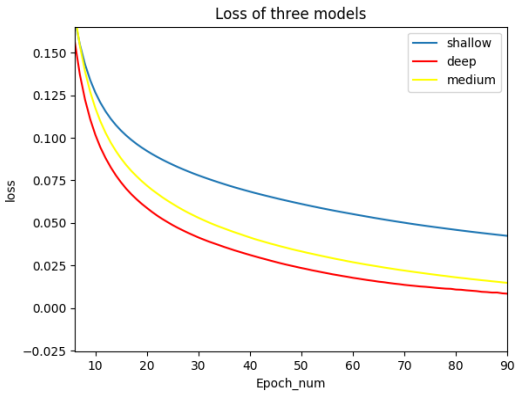
dense_2 : Dense	input :	(None, 9)
	output :	(None, 10)

Total = 8725

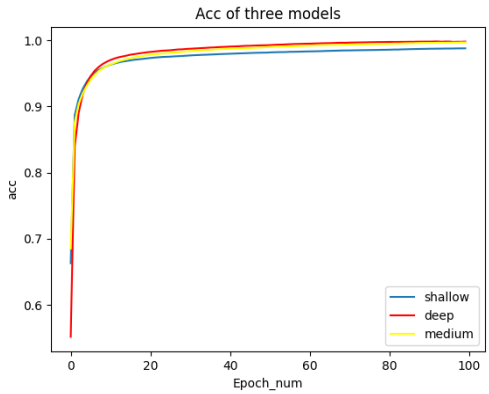
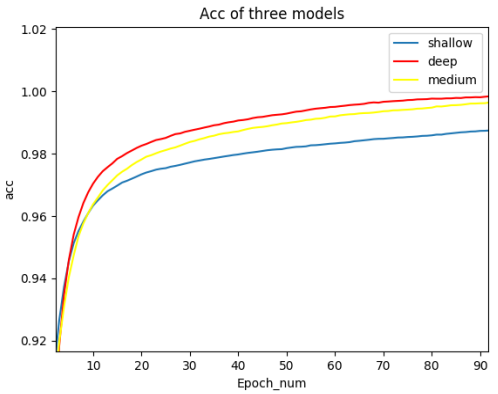
從左到右命名三個model為 **shallow**, **medium**, 和 **deep** ，各有 **8859**, **8579**, 和 **8725** 個參數.

Activation function : **relu**

三個model的loss

Loss	Zoom in
	

三個model的正確率

Accuracy	Zoom in
	

Comment

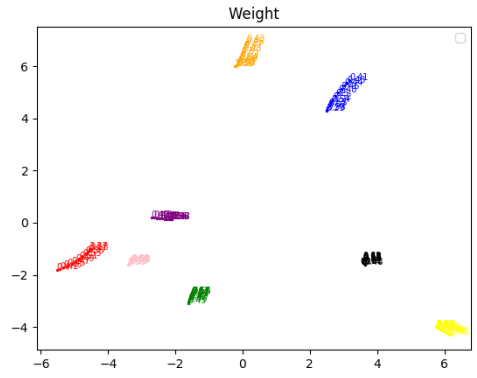
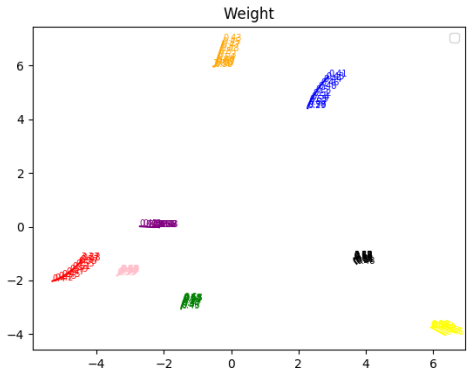
- 在loss和accuracy上，都是deep做得比median好，median做得比shallow好，結論是越深的模型在MNIST上面做得越好．

1-2-1 Visualize the Optimization Process

模型設定

- 訓練在目標函數：$y = \cos^2(x)(\sum_{i=1}^{100} \frac{2\sin(ix)}{(-1)^i i\pi} + \sum_{j=1}^{100} \frac{2\sin(\frac{jx}{2})}{(-1)^j \frac{j\pi}{2}})$
- DNN(input : 1, output : 100) -> Relu -> DNN(input : 100, output : 1)
- 先用上面的模型訓練8次，每次1000個epochs，先記錄每個epoch整個模型的weight和第二層的weight，所以最後會有8000個整個模型的weight和8000個第二層的weight，兩個分別丟到PCA降到二維，最後再分成8組資料每組有1000個分別作圖（一組1000個每80個畫1個到圖上）．

Plot (Note : 下圖的點都是往外擴散，圖上的數字是loss)

整個模型的weight	layer 2
	

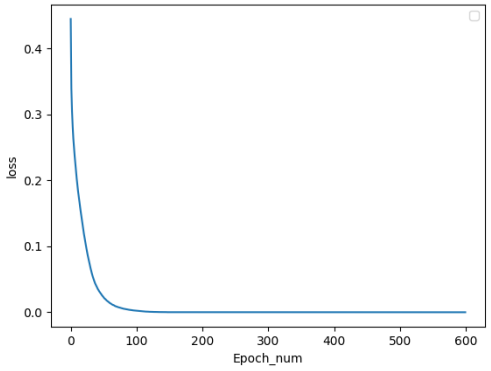
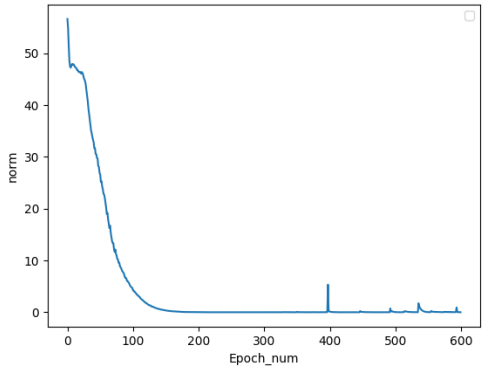
Comments

- 先看整個模型的圖，每個點上的數字是loss所以圖上的每次訓練都是往外擴展的，而且趨勢是一開始的梯度是小的、比較平滑的，訓練到後面每次weight的移動越多，梯度比較大．
- 再來比較整個的和layer2的可以看出來在這樣比較小的模型上，layer2的weight跟整個模型的weight趨勢差不多，原因是layer2的參數跟整個模型的參數比例夠高，layer2的weight趨勢能夠表示整個模型．

1-2-2 Observe gradient norm during training

- 訓練在目標函數：$y = \cos^2(x)(\sum_{i=1}^{100} \frac{2\sin(ix)}{(-1)^i i\pi} + \sum_{j=1}^{100} \frac{2\sin(\frac{jx}{2})}{(-1)^j \frac{j\pi}{2}})$, 五層DNN

Plot

Loss	Gradient norm
	

Comment

- 由圖上可以看出在loss越小的同時，gradient norm越小，代表模型的參數到了越接近0的地方．

1-2-3 What happens when gradient is almost zero?

- 訓練在目標函數：$y = x^2$，參數總量31，兩層DNN

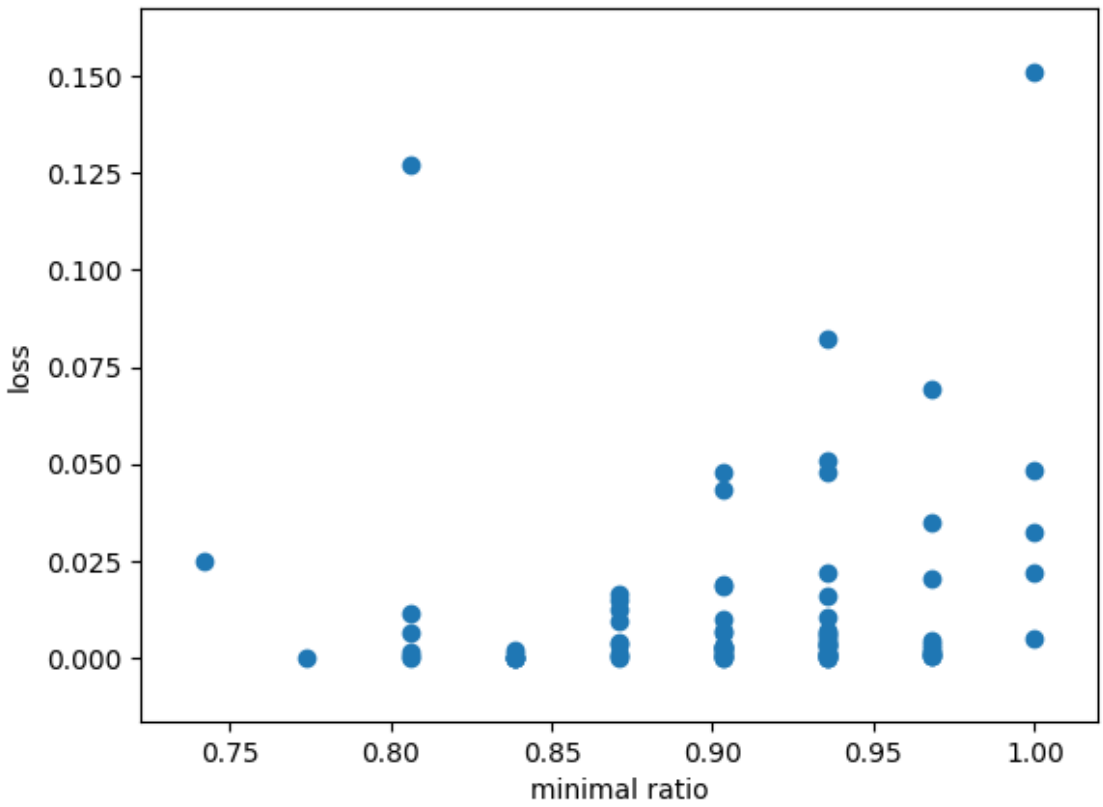
讓gradient norm接近0

- Optimizer 先用Adam訓練一段時間，最後把optimizer改成LBFGS(second derivative)，當norm降到小於0.1的時候紀錄這時的minimal ratio，然後再做下一次訓練．

算minimal ratio

- 拿到當時的weight後，算出hessian matrix，改成對角矩陣，拿去算eigenvalue，拿到不超過31個值，最後以大於零的值的比例算minimal ratio．

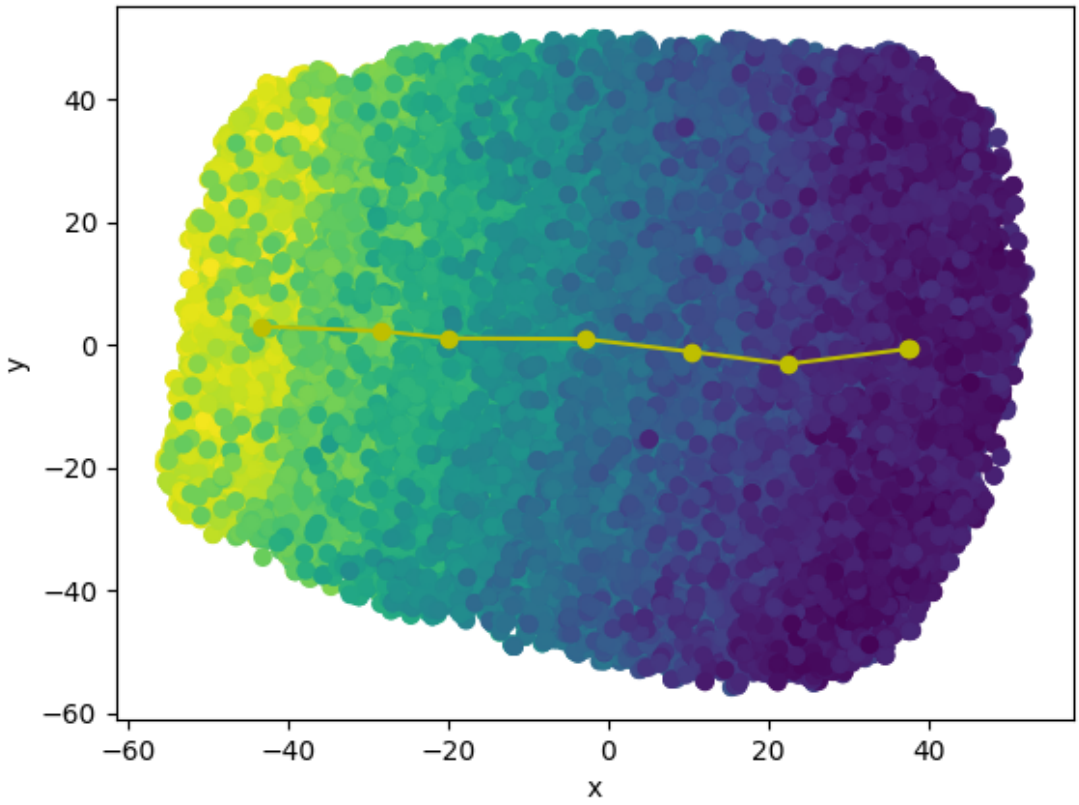
Plot



Comments

- 圖上可以稍稍看出當loss越低，minimal ratio越高的趨勢，不過有些點的loss雖然是0，但是minimal ratio沒有比較高，原因可能是選到的點loss都夠低，所以minimal ratio的值大於0.75都算滿高的．

1-2-bonus Visualize the error surface



- 圖上中間的線是像1-2-1的做法得出的線，是從左到右的趨勢，線外面的每個點都是隨機打點的結果，顏色代表該點loss的大小，可以看出越右邊的loss比較低，左邊的比較高
- 做法是用小的網路架構去訓練簡單的函數(y=x^2)，在訓練到一半的時候，把接下來8個epoch的weight記錄下來，每個weight再對周圍打4000個點(對每一個parameter加上一個隨機很小的數字)，最後會有8*4000個隨機打點和8個模型在訓練時走過的點，一起先丟到PCA降到10維，再把結果丟到TSNE降到2維．

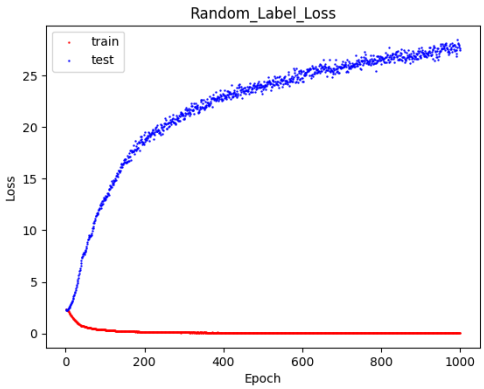
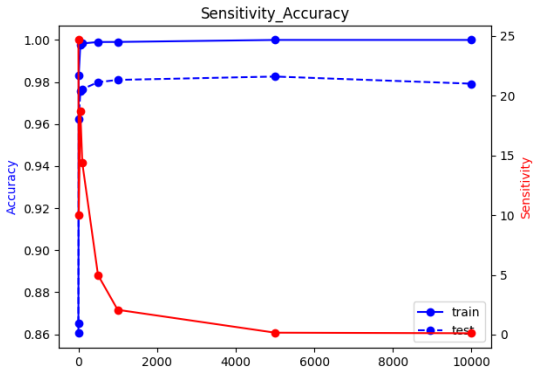
1-3-1 Can network fit random variables?

設定和實驗

MNIST

```
Hidden layer = (700,700,700),第一二中間用relu。
Loss = Cross-Entropy, Optimizer = Adam, Learning Rate = 1e-4, Batch size =
100
```

Plot

Loss	Accuracy
	

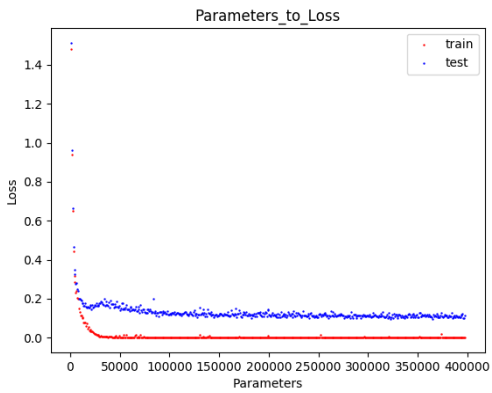
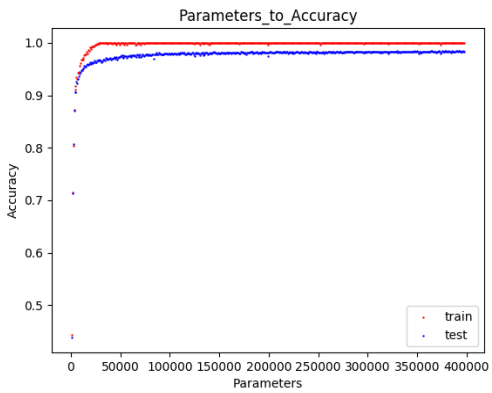
1-3-2 Number of parameters v.s. Generalization

設定和實驗

MNIST

```
Hidden layer = (700,1,2) ~ (700,500,1000), 第一二中間用relu。
Loss = Cross-Entropy, Optimizer = Adam, Learning Rate = 1e-3, Batch size =
100
```

Plot

Loss	Accuracy
	

Comments

- 當參數逐漸增加,loss一開始快速下降，在參數約25000的時候達到穩定態。Accuracy則是逐漸升高，趨勢和loss差不多。這次的實驗應證越複雜的網路越有能力學會目標。

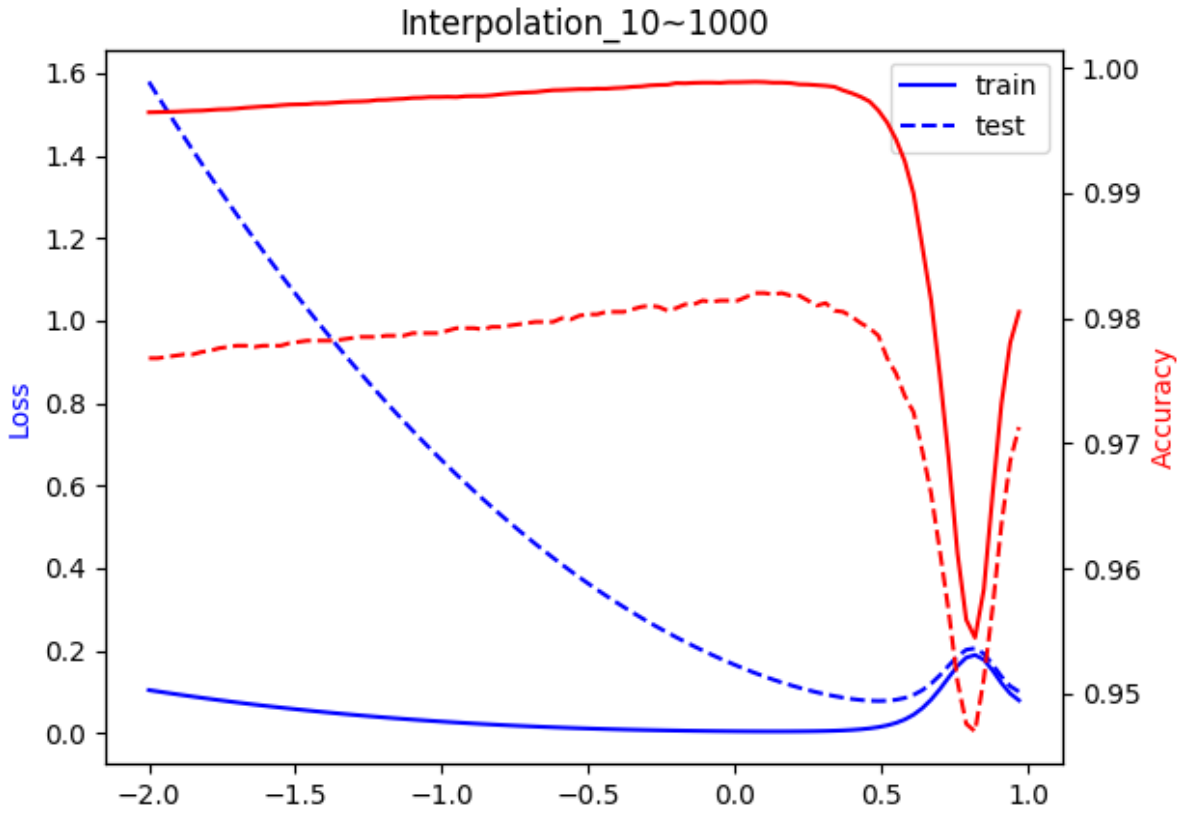
1-3-3 Flatness v.s. Generalization Part-1

設定和實驗

MNIST

```
Hidden layer = (500,500), 第一二中間用relu。
Loss = Cross-Entropy, Optimizer = Adam, Learning Rate = 1e-3, Batch size =
50 ~ 1000, alpha = -2 ~ 1
```

Plot



Comments

- 在大約alpha =1和0的時候整體結果最好，這也證明在訓練時，即使兩組收斂的網路，其對應neuron的意義也不相同。因此組合兩者的參數幾乎必定會使結果變差。

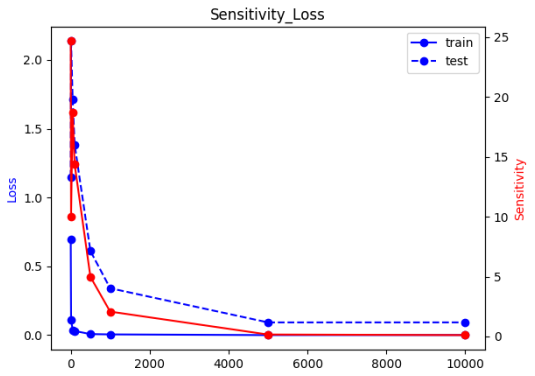
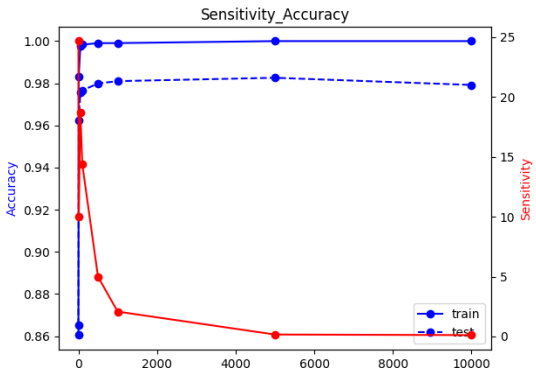
1-3-3 Flatness v.s. Generalization Part-2

設定和實驗

MNIST

```
Hidden layer = (500,500), 第一二中間用relu。
Loss = Cross-Entropy, Optimizer = Adam, Learning Rate = 1e-3, Batch size =
1,10,50,100,500,1000,5000,10000
```

Plot

Loss	Accuracy
	

Comments

- 當batch size越小，其對於資料的敏感度越高。顯示在大的batch中，w的gradient　因為受不同資料的交互影響而有抵銷的現象，反而不如一次看一筆來得敏感。

1-3-3-bonus Flatness v.s. Generalization Part-3

- 如下表，我們用兩層DNN訓練MNIST數次，用6個不同的batch size，在每次訓練，先訓練一段時間後，在目前模型參數所在的位置附近隨機打150個點，算這些點的平均loss，最後算平均loss跟目前訓練的loss的差距得出loss range

Batch size	Sensitivity	Loss range
50	705.35145	0.02462
100	859.82603	0.01442
500	224.44220	0.00295
1000	87.98782	0.00329
5000	15.77452	0.00438
10000	13.74752	0.01703

- 依理論，batch size越大，sensitivity越小，會落在越陡峭的地方，所以loss range會越大，這在batch size 500~10000從表上可以看得出來，但是在batch size 50和100的地方，loss range也是很大，推測可能是sensitivity太高，沒有一個地方平坦到可以侷限住它，所以再算loss range的時候，它就不小心跳到比較陡峭的地方了．

分工表

	B05902031	**B05902008**
1-1-1	測試參數＋報告	模板
1–1-2	測試參數＋報告	模板
1-2-1	測試＋報告	模板
1-2-2	測試＋報告	模板
1-2-3	全	無
1-2-bonus	全	無
1-3	無	全
1-3-bonus	全	無