Neural Network

1-hidden layer

[activation function] : sigmoid function

[gradient descent] : stochastic gradient descent

以 Traing：Validation = 9：1 的比例做資料切割。 (例如：第一次，將照片編號尾數為 0 的作為 Validation data，其餘為 Training data … 之後以此類推)

一開始把 learning rate η 設定為 1，發現設太高，weight 修正太多，導致分類表現沒有很好，之後將 η 降為 0.01，並且多做幾次 error back-propagation 後，得到的分類結果還不錯。接著嘗試不同 nodes 數、不同 η 值、做 error back-propagation 的次數 (所謂“一次”表示：對所有 training data 都做一次 stochastic gradient descent，以下簡稱 #EBP，) 所訓練出來的模型，其 validation data 的錯誤率(error rate)，並整理成下表

nodes	*η*	*# EBP*	*Error rate*
5	1	1	0.666
		2	0.866
		5	0.153
		10	0.236
	0.1	1	0.226
		2	0.333
		5	0.333
		10	0.083
	0.01	1	0.333
		2	0.043
		5	0.003

		10	0
20	0.01	1	0.173
		2	0.003
		5	0.006
		10	0.006
50	0.01	1	0
		2	0
		5	0.003
		10	0.003
100	0.01	1	0
		2	0
		5	0
		10	0

觀察上表，可以得到以下結論：

(1). Nodes 數量越多，model 表現越好(太多可能會 overfitting?)。

(2). η 不能設太高，否則 weight 一次會調整太多；也不能設太小，不然會算很久，更慘的是可能只能找到 Local min。在這邊發現設為 0.01 表現還不錯，大約五次以內就會收斂。

此外，以 $\text{nodes} = 10,\ \eta = 0.01,\ \#\text{EBP} = 10$ 的條件做 cross-validation 的時候，發現有幾個 case 沒辦法收斂，但有收斂的話，error rate 幾乎都為 0，這部分將在 2-hidden layer 那邊討論。

2-hidden layer

[activation function] : rectified function $f(x) = \max\{0, x\}$

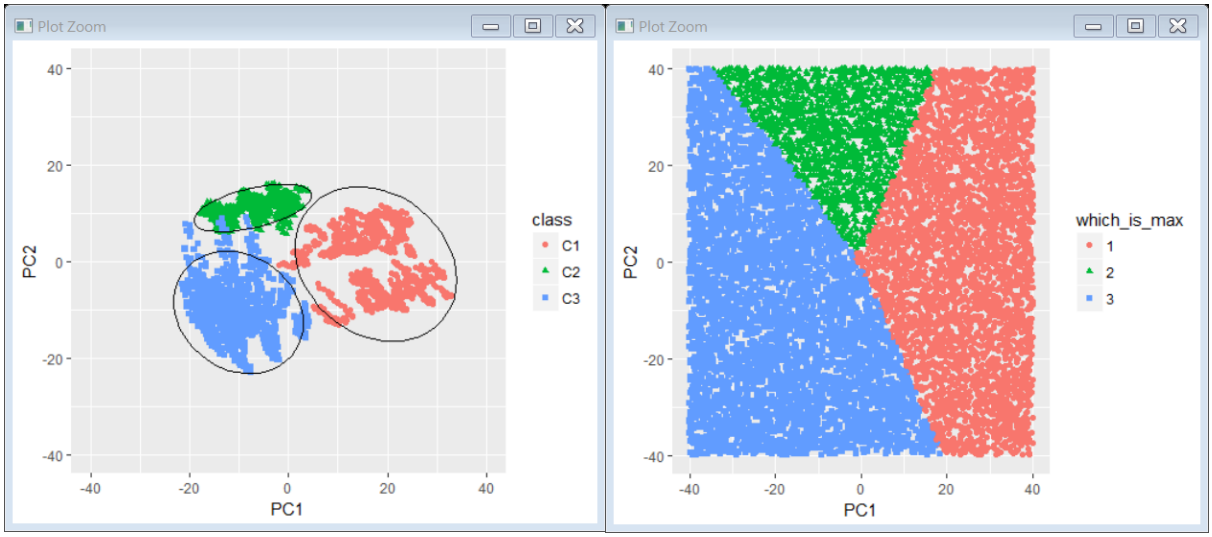
[gradient descent] : stochastic gradient descent

與上題不同的地方，除了多了一層 layer 之外，還以 rectified function 取代平滑的 sigmoid function，如此一來可以提高收斂速度。接著以 $\text{nodes } 1 = \text{nodes } 2 = 10,\ \eta = 0.01$ 的條件做 cross-validation

Cross validation 尾數	*Error rate*
0	0.003
1	0.333
2	0.226
3	0
4	0.003
5	0
6	0.333
7	0.243
8	0.033
9	0.006

由上表可以發現，並不是所有的 case 都會收斂，常常會有其中一類全部被分到另一類（例如：C2 全部被分到 C3），使得錯誤率為 0.333。猜想可能因為資料數不足，硬要用兩層 Neural network，所以導致效果不穩定；也可能是因為 C2、C3 本身就沒分得很開，所以透過提高 PCA 的維度，讓整體解釋變異量佔比提高，或許會有幫助。

Decision regions



左圖為 training data 在 PC1、PC2 上的分布情形。

以 uniform distribution 在 PC1、PC2 上隨機產生一萬個點，再乘上 training data 所得的 weight，得到右圖的 decision region。

Compare to hw2

相較於 HW2 的 model，這邊受 initial weight 影響很大，如果運氣不好，常常沒辦法收斂，但只要收斂的話，準確率比 HW2 的好很多(幾乎達到 0 error)。

此外，η 的大小也影響很大，若太大的話，一次會調整太多，導致分類結果不好；但太小的話，weight 幾乎調不太動，除了要算很多次之外，還可能只能找到 local min。

Nouns explaination

✧ Batch gradient descent（以前作業用的方法）

一口氣把全部的 training data 丟進去跑，再調整 weight。優點是省時間，因為只要調一次 weight，但相對缺點為計算複雜度會提高。

✧ Stochastic gradient descent（本次作業的方法）

一次只丟一筆 training data，並更新一次 weight，再做下一筆 data。優點為計算簡單，但由於有Ｎ筆 data 就要算Ｎ次，所以在樣本數多的情況下，效率可能不會很好，且收斂不太穩定。

✧ Mini-batch gradient descent

將 training data 分割成數個獨立子集合，然後一次丟一個子集合下去跑，跑完後取平均，再做一次調整：接著再丟第二個子集合…以此類推，所以假設原本有 10000 筆 data，若分成 100 個獨立的子集合，就只要調整 100 次 weight，提升效率。mini-batch 結合上述兩種方法的優缺點，實務上的表現還不錯。

✧ Online gradient descent

與ＳＧＤ很像，也是一筆一筆資料下去跑，但差別在於，前面三種方法的 training data set 都是固定的，online 顧名思義就是即時更新，一有新的 data 送進來，我就即時更新一次 model。

（就好像股票一樣，你的 dataset 不是固定的，每分每秒都會有新的資料進來）