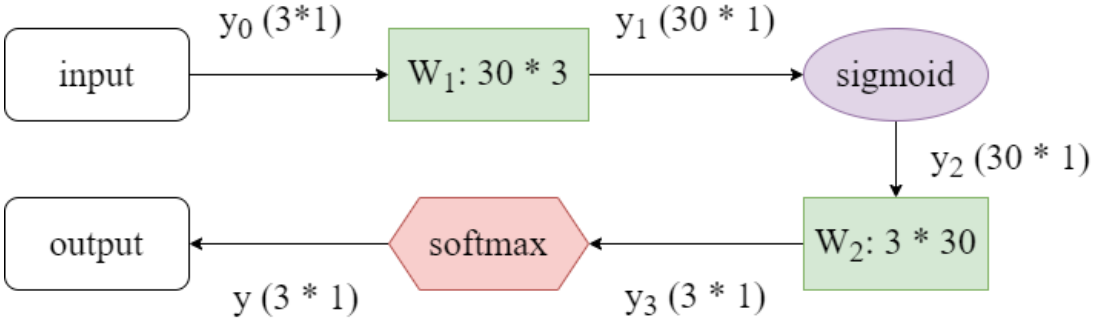
ML HW3

107061218 謝霖泳

1. model architecture and PCA method

- model architecture (2 layer NN)



我將hidden neuron的數量設為30，並依照作業要求以`sigmoid()`作為non-linear。

input端的大小為3*1，前兩個element就是每張圖做完PCA後的兩個數字，而最後一個是bias，我將bias一起放在input中，一起和`W`做矩陣相乘，這樣一來就不用再相乘之後還要多一個加bias的動作，更重要的是，這樣在做back propagation時，便只需要update `W`，不用計算bias的更新，計算上方便許多。

而在計算loss的部分，我使用的是**cross entropy loss**，得到loss之後可以進行back propagation，這時候要計算gradient，而根據back propagation的規則，可以得到以下關係式。其中，第三式的@為element-wise的相乘，而`sigmoid'()`代表sigmoid函數的微分，即`sigmoid'(t) = sigmoid(t)*(1 - sigmoid(t))`。

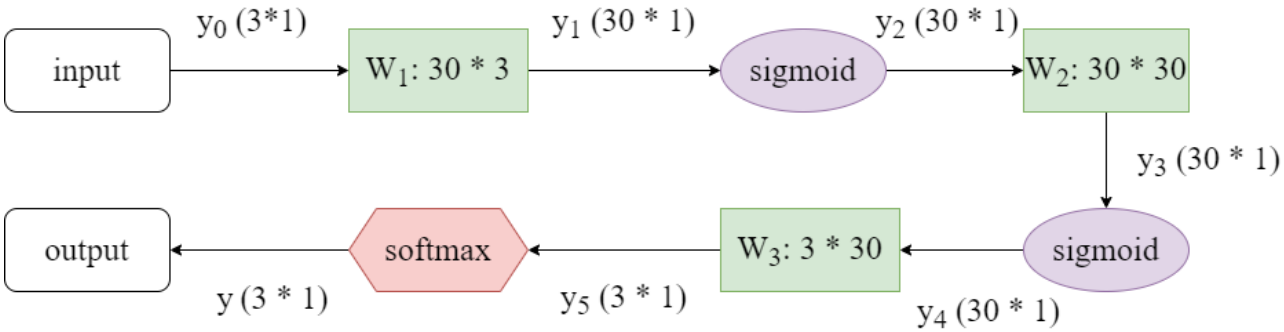
$$d_{y\ (3*1)} = Loss_{\ (3*1)}$$

$$d_{y2\ (300*1)} = W_2^{T}{}_{\ (300*3)}\ d_{y\ (3*1)}$$

$$d_{y1\ (300*1)} = d_{y2\ (300*1)} \ @\ sigmoid'(y1)_{\ (300*1)}$$

$$d_{W1\ (300*3)} = d_{y1\ (300*1)}\ y_0^{T}{}_{\ (1*3)}$$

- model architecture (3 layer NN)



一樣以每層的hidden neuron數量=30為例，3 layer neural network的結構如上，基本上和2 layer類似，只是中間再多一層而已，唯一不同的就是weight的size，目的是要確保每層layer中的`W`和`x`是可乘的。而3 layer NN的back propagation gradient計算基本上跟2 layer的情況差不多，因此不再贅述。

- PCA method

  PCA我使用`sklearn`已經內建好的PCA function，將每張32*32的圖片讀進來存成一維陣列以後，呼叫`pca.fit_transform()`去fit and transform training set，並將validation set及testing set做`pca.transform()`。

  我一開始在這邊犯了一個錯誤，就是將三種水果都各自fit and transform，結果train了半天accuracy永遠都在40幾%，僅僅比亂猜好一點而已，經過好心同學的提點才知道原來`pca.fit_transform()`和`pca.transform()`不能隨便亂用會出事，應該要向我下方這段code這樣，先選定一個基準讓PCA fit它，並transform出所需的參數，而其他的data就應該要依照剛剛fit的這個標準進行transform，而不是各自都重新fit一遍，感謝熱心同學的提點。

```
pca = PCA(n_components=2)
training_set = pca.fit_transform(training_set)
vld_set = pca.transform(vld_set)
testing_set = pca.transform(testing_set)
```

2. test accuracy

- 不同hidden unit數對test accuracy的影響 (epoch=`200`, learning rate=`1e-4`)

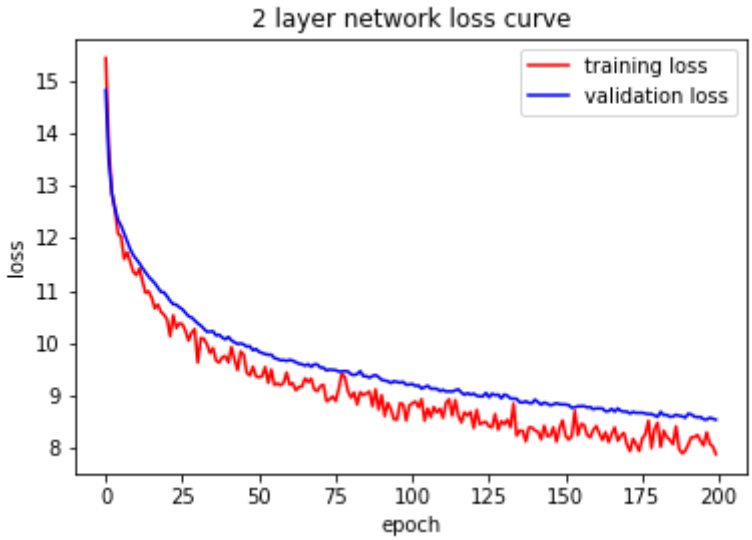
number of hidden units	**5**	**10**	**30**	**100**
2 layer accuracy	0.58	0.92	0.80	0.71
3 layer accuracy	0.63	0.73	0.78	0.76

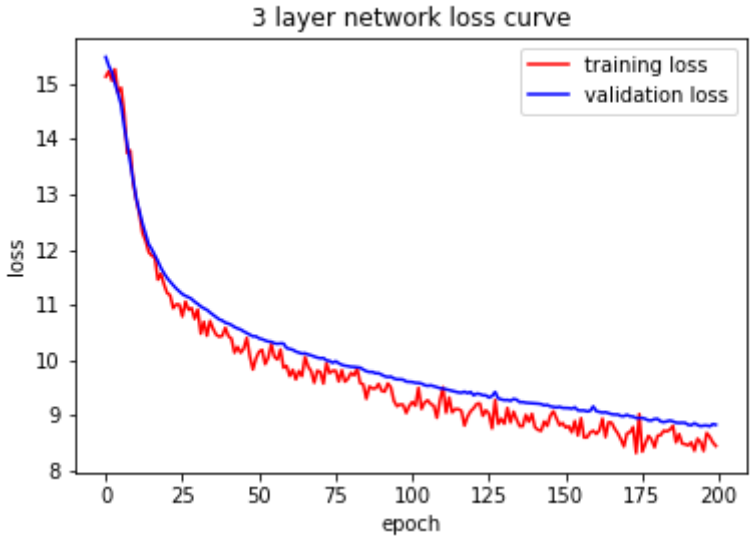
- 不同learning rate數對test accuracy的影響 (epoch=`200`, number of neuron = 50)

learning rate	**1e-1**	**1e-2**	**1e-4**	**1e-7**
2 layer accuracy	0.71	0.79	0.83	0.39
3 layer accuracy	0.79	0.73	0.78	0.33

3. training loss curves (cross entropy loss)

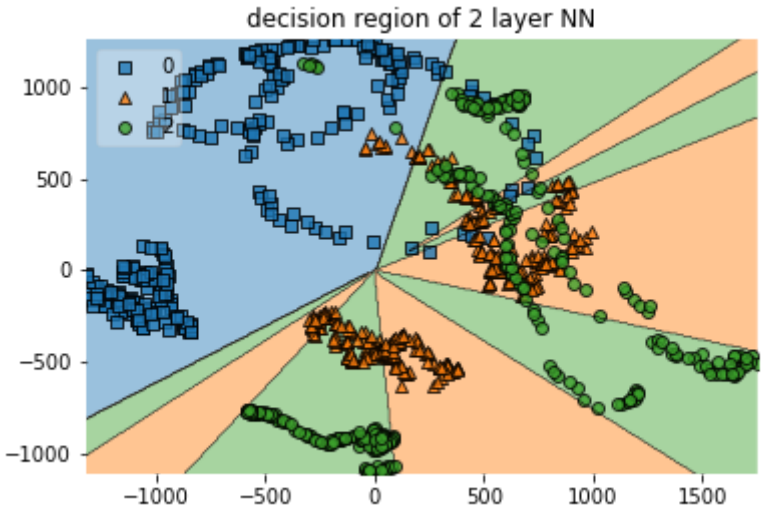
- # of hidden units = 5, epoch = 200, learning = 1e-4

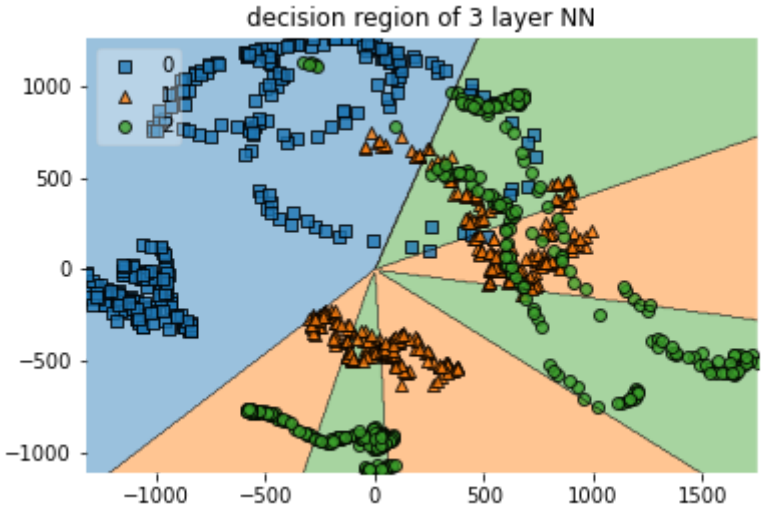




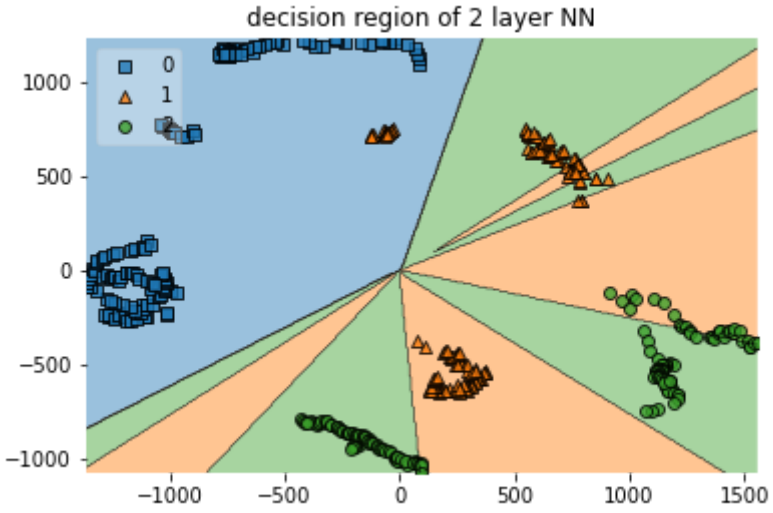
4. decision regions

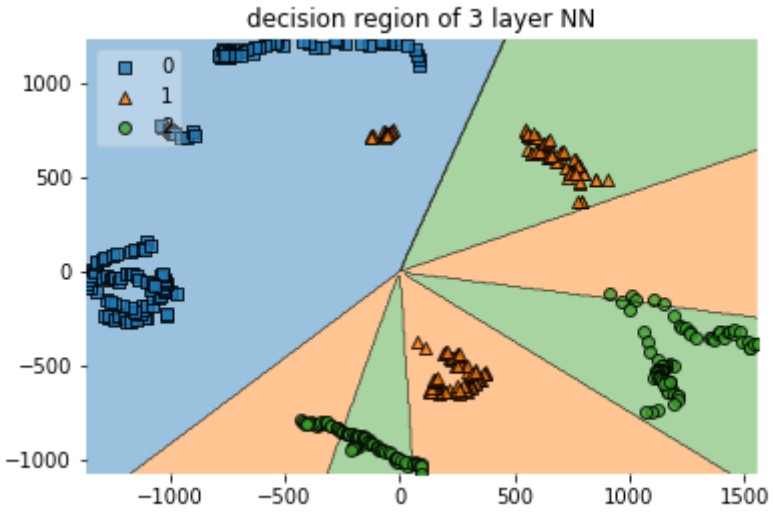
- decision regions of training set





- decision regions of testing set





5. discussion

在test accuracy中，可以發現當neuron數增加時，對於accuracy的提升是有幫助的，但觀察hidden unit = 100時，不管是2 layer NN或3 layer NN的表現都較hidden unit = 30還要差，原因是因為neuron數太多時，對data容易產生overfitting，因此導致test accuracy較低。

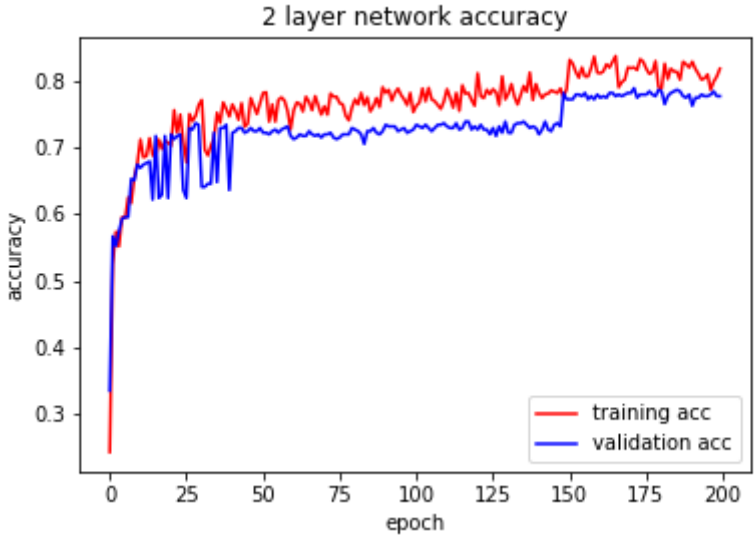
再來觀察test accuracy的第二個表格，可以發現當我的learning rate小到只剩1e-7時，test accuracy非常低，約為1/3，就跟隨便亂猜差不多，代表model根本沒有東西，所以將learning rate逐漸調大的過程中，可以發現accuracy漸漸上升，甚至可以達到八成以上的準確率，對於一個結構如此簡單的network來說是非常值得讚嘆的。

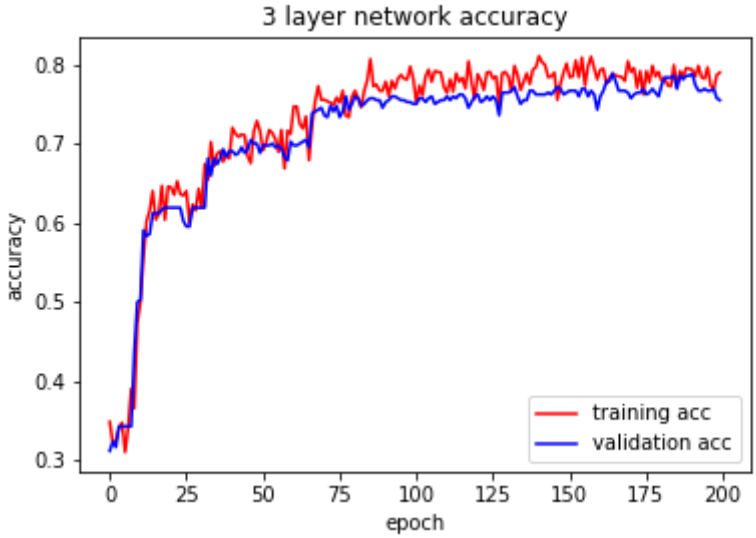
再來，當learning rate繼續升高到0.1時，accuracy又再次掉了下來，原因是因為learning rate決定了gradient descent每一次更新W應該要更新多大的幅度。如果learning rate太高，代表我們的腳步太大，可能走一步就已經跨越了好幾個local minimum，因此當然走不到最低點。

最後是有關decision region的觀察，這邊我只放出training set和testing set的decision region，而validation set的decision region基本上分布得跟training set差不多，因為它本來就是從training se被切分出來的。在這兩張圖中可以發現，我的背景雖然都是分成三種顏色，但卻是被切分成了將近10塊區域，原因我推測是因為我沒有對data做PCA以外的前處理，比如normalization，因此data的分布較容易受極端質影響，所以背景才會被切成好多塊。

另一個值得注意的點是從上面兩組decision region的圖可以明顯發現，training data的分佈相較於testing data非常的分散，被切分成更多塊，而testing data的分佈就相對很集中，因此在作作業的過程中常常發現testing的accuracy常常都比training或validation accuracy還要大一些。從這個經驗也讓我學到，往後在作ML相關研究，在準備資料的時候應該避免這種情形，應該要讓training, validation及testing set的分佈都接近一點會比較好。

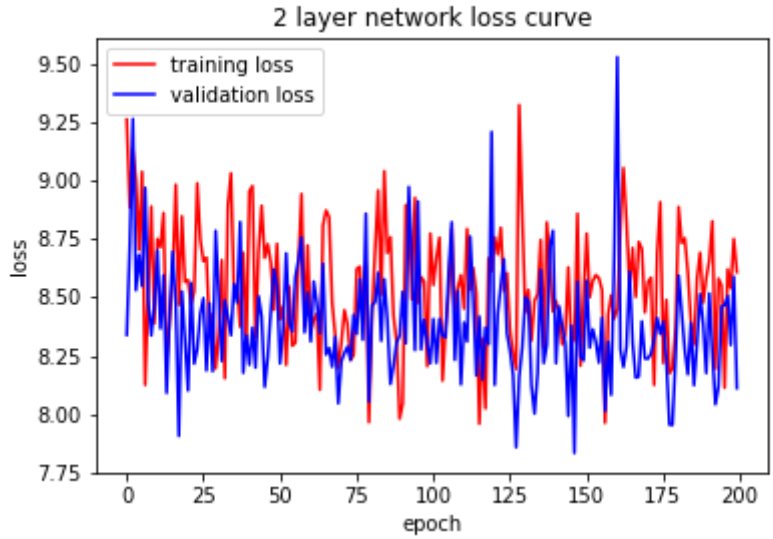
此外，我有將accuracy的圖一併畫出來，以# of hidden units = 50, epoch = 200, learning = 1e-4為例，得到以下的learning curve。

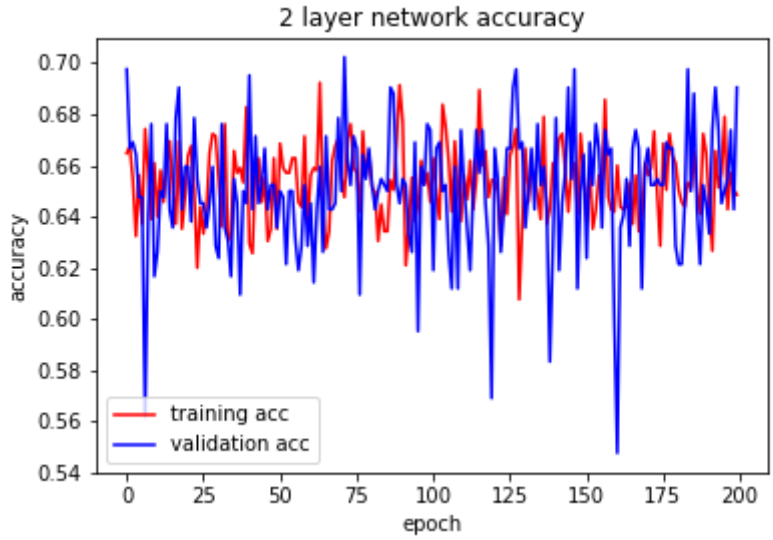




從以上的learning curve可以發現，accuracy從0.3左右逐漸上升到約0.8，代表更新weight這件事情是真的有做對，而中間會出現上下小幅波動就是因為我們用的gradient descent方法是Stochastic gradient descent, Stochastic本身就帶有隨機性在裡面，這樣才不容易在training的過程中對資料產生overfitting。

當learning rate太高時(在這裡將learning rate設為0.1)，就會得到像下圖的結果，可以發現loss沒有像上方的圖一樣有收斂的趨勢，而是持續震盪，在這種model根本沒有好好學到東西的情況下，從accuracy的圖也可以看出來，沒有漸漸變高而是不停在0.65附近震盪。





最後，總結一下2 layer跟3 layer對於本次作業的影響，我認為layer數本身在這次作業中並沒有太大影響，從上面所有的例子我們都可以發現2 layer跟3 layer的表現好壞與趨勢基本上大同小異，我認為原因是因為input data太簡單，class數也太少，因此只需要少少的neuron就能達到不錯的效果，因此增加成3 layer並沒有對accuracy產生太多的improvement，反而有時候2 layer的表現還比3 layer更好，因為在data如此少又簡單的條件下，neuron一多產生overfitting的風險就會變高。因此，若要看出layer對於結果的明顯影響，我認為可能要到複雜一點的case上才看得出來。