Deep Learning HW1

0756708 孫茂勛

【作業描述】

實作 Deep Neural Network(DNN)，並且不能使用可以自動微分的框架(Pytorch, Tensorflow, Keras)，最後回答各個小題。

【註記】

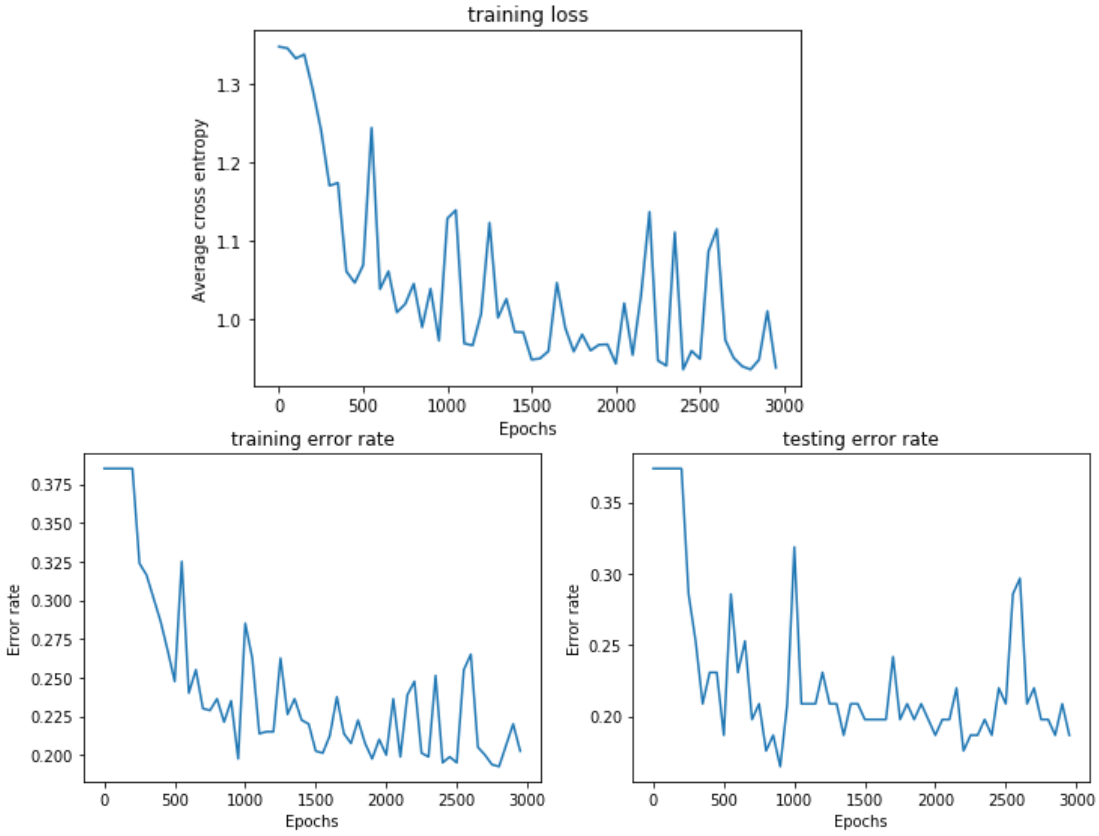
由於我是使用 jupyter 完成，不太好直接執行.py 檔案(我目前把第一題以外的註解掉了)，所以我有另外附一份.html 檔案，是 jupyter 內各小題的執行結果。

【Problem 1】

設計一個 DNN 架構 for binary classification，loss 使用 cross entropy，使用 mini-batch + SGD 來更新參數，然後介紹自己的架構以及為何這樣設計。

這次的作業我透過 Class 來實作一個 DNN 架構，之後只要指定各層的 neuron 數量、batch size 和 learning rate 就可以自動訓練。以下是我所使用的架構參數和實驗結果(learning curve、training error rate 和 test error rate)。

Model architecture	
Neuron of each layers	[6, 32, 32, 64, 2]
Epoch(Number of iteration)	3000
Batch size	100
Learning rate	0.3



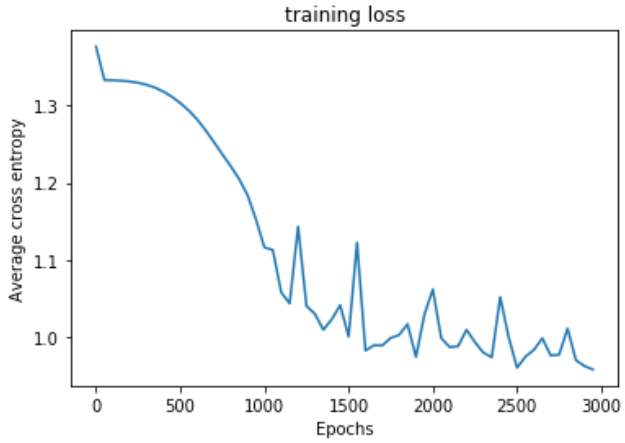
之所以這樣設置的原因是因為我其實是從第二題(指定 layer 的 neuron 數量)先開始寫，而在第二題中達到了不錯的效果，於是我想嘗試使不同 neurons 的數量多一點，並改變 learning rate(太小似乎會跑很久，因為比較多層)，看能否透過提升模型的複雜度來持續降低 error rate，實驗結果顯示能夠較快的收斂(約在 200 多 Epochs 時後 loss 和 error rate 就開始往下降)。

【Problem 2】

建立一個 DNN 並且個層的 neuron 數量為[6 3 3 2]。

實驗參數和結果如下：

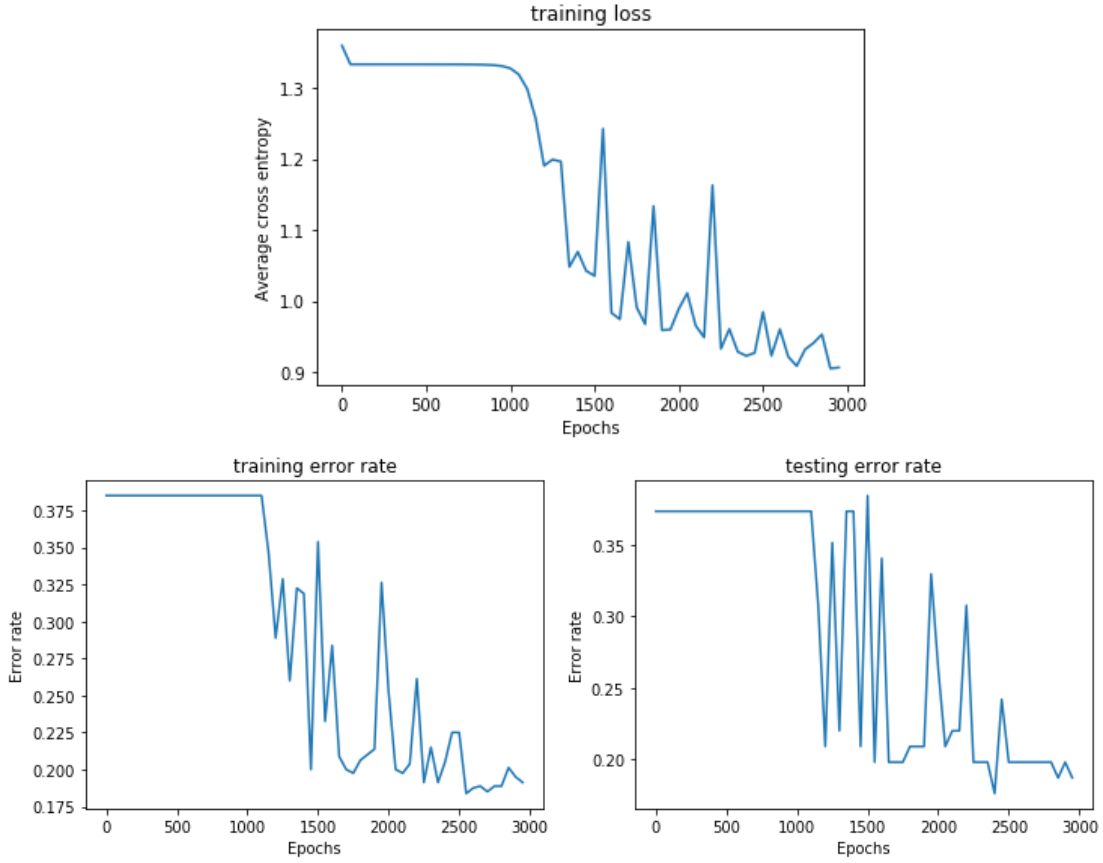
Model architecture	
Neuron of each layers	[6, 3, 3, 2]
Epoch(Number of iteration)	3000
Batch size	100
Learning rate	0.03



【Problem 3】

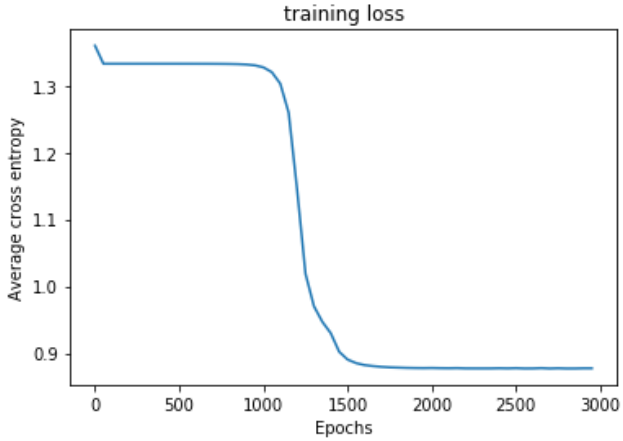
將 `Fare`這個 feature 做 Normalization，比較結果並且回答是否有其他需要做 Normalization 的 Feature?

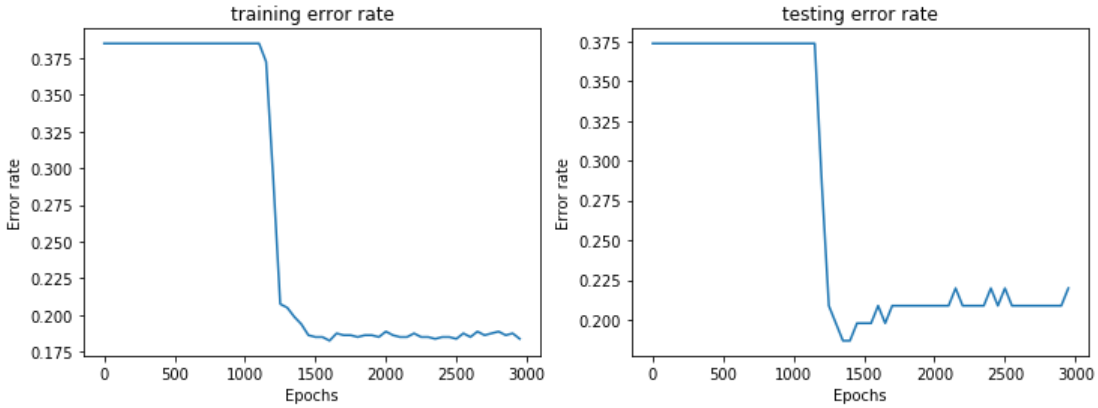
透過 sklearn 的 StandardScaler 對`Fare`做了 Normalization，使其 range 變成平均值為 0，標準差為 1。套用 Problem2 一樣的參數後發現在 Epoch 相同的情況下，loss cruve 很難下降，推測是做了 Normalization 後數值變化較小，所以需要稍微調大一點 Learning rate，於是我將 learning rate 調至 0.1，實驗結果如下圖所示，做了 Normalization 後的 training loss 以及 error rate 都有較佳的結果 (雖然圖片不是很明顯，不過數值上的確有差異)。



我認為原因是因為`Fare`是連續型的 feature，並且彼此落差都很大，在 training 的時候很容易因為突然遇到一個很大的值造成 gradient 有很大的更新，透過 Normalization 則可以把值的範圍縮小到一個區間內，減少 loss cruve 的震盪。同樣的方法也適用於連續型的 feature `Age`，下圖是也將`Age`做 Normalization 後的結果(參數參照 Problem 2)，從圖中可以發現訓練過程收斂的更加穩定，不太會有大幅的震盪，而且結果也會比較好。

最後，因為`SibSp`和`Parch`的值本來就不大，我認為就不太需要做這個動作。此外，離散型的特徵`Pclass`和`Sex`則不需要做這個動作。





【Problem 4】

將找出對 performance 影響最大的 feature，並描述如何找出的。

觀察第一題訓練出來的 module weights，其中發現最大的 weights value 是`Age`這個 feature，代表再做 y = wx+b 時，`Age`的值將會有最大的影響力。事實上，當初鐵達尼號沈船時，第一優先上避難船的是女性。值得一提的是，第二大的 weight 是`Pclass`這個 feature。

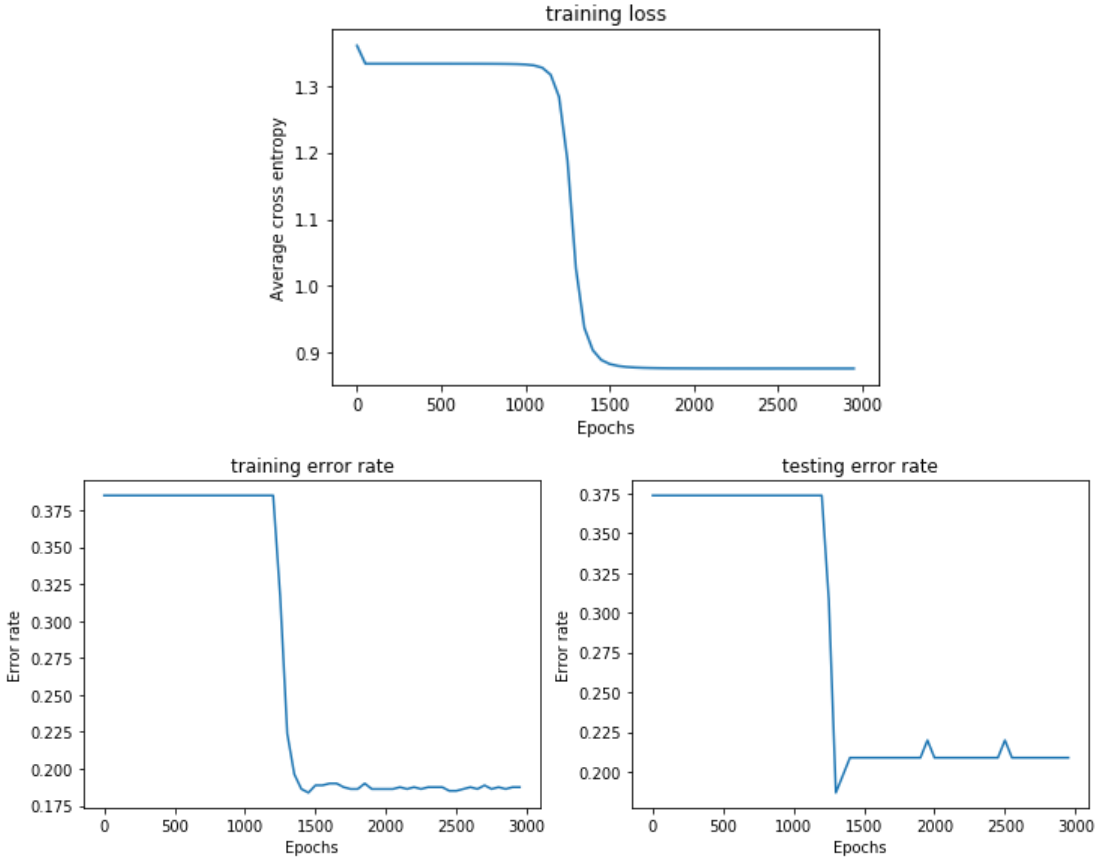
```
In [18]: module2.weights[0]
         executed in 5ms, finished 20:44:45 2019-04-08
Out[18]: array([[ 0.4874818 ,  4.40971608,  0.01587691,  0.69172897,  0.33305862,
                  -0.08210872],
                [ 0.4874818 ,  4.40971608,  0.01587691,  0.69172897,  0.33305862,
                  -0.08210872],
                [ 0.4874818 ,  4.40971608,  0.01587691,  0.69172897,  0.33305862,
                  -0.08210872]])
```

Pclass	Sex	Age	SibSp	Parch	Fare
3	1	22.0	1	0	7.2500
1	0	38.0	1	0	71.2833
3	0	26.0	0	0	7.9250
1	0	35.0	1	0	53.1000
3	1	35.0	0	0	8.0500

【Problem 5】

解釋是否應該要對`ticket class`做 one-hot?

我覺得應該需要，由於`ticket class`是 categorical feature，其實數字之間的差異是沒有關係的，比方說 3 和 1 之間的差距比起 3 和 2 多了 1，可是實際上只是代表他們是不同的 class，這個差距不應該被當作預測時的一個依據，如果沒有做 one-hot encoding，有可能會造成這個距離的關係也被納入考慮。下圖是做 one-hot 後的訓練結果(參數和資料處理接續 Problem 3)。



【Problem 6】

設計兩個 new samples 一個會死翹翹一個不會。囧。

苦主 1 號: 就是我本人啦 QQ

即將被預測死掉的受害者資訊	
Pclass	3
Sex	1
Age	23
SibSp	2
Parch	2
Fare	0.87

```
In [63]: prediction_john = module5.forward(people_John)
         print('John死亡的機率vs存活的機率:', prediction_john[0], prediction_john[1] )
         executed in 6ms, finished 21:44:23 2019-04-08
         John死亡的機率vs存活的機率: [0.87382879] [0.12617121]
```

我有 0.87 的機率會死掉 QQ，可憐的孩子。

幸運者 1 號(我掰的): 她叫做 Angela，18 歲~~，很正~~，非常有錢，有一個哥哥，跟爸爸媽媽一起出來玩。

即將被預測生存的幸運者者資訊	
Pclass	1
Sex	0
Age	18
SibSp	1
Parch	2
Fare	20

```
In [65]: prediction_Angela = module5.forward(people_Angela)
         print('Angela死亡的機率vs存活的機率:', prediction_Angela[0], prediction_Angela[1] )
         executed in 5ms, finished 21:44:51 2019-04-08
         Angela死亡的機率vs存活的機率: [0.03808093] [0.96191907]
```

Angela 有 0.96 的機率會存活下來，~~人正真好(咦)。~~

為什麼會這樣勒，其實鐵達尼號當初婦女、小孩是優先上逃生船的，以及你在不同的船艙(價位不同)也會多少有影響(https://kknews.cc/zh-tw/history/lzxnq8e.html)，所以只要是女生基本上生存機率就會比較大。
所以苦主一號(就是我啦 QQ)我選擇了男生，並且很貧窮的設定；而幸運者 Angela 則選擇了富家千金。

【心得】

我不知道要不要附啦，不過說不定有附有加分(咦)，就寫一下下...

這次的作業最難的地方應該就是在實作 back propagation 了，以往我在寫 DNN 都是 pytorch、keras 套一套，對於內部的細節都是似懂非懂。要完成這次作業必須更清楚 weights 在每一層 layers 之間的傳遞與反傳遞過程，如果沒有搞懂的話是沒辦法完成這次作業的，所以一開始寫的時候極度崩潰。
最後在網路上找到一篇教學網站(http://neuralnetworksanddeeplearning.com)，覺得內容寫的很好，將該網站的內容仔細的讀過一遍，並把 back propagation 搞懂了之後自己把 code 寫出來，最後做完作業之後感覺自己對於 back propagation 了解的更加清楚了。也謝謝過程中助教很熱心地回答我的問題，讓我能夠順利完成這次作業。