DLP – LAB02: Back Propagation

學號：310611008 姓名：張祐誠

Introduction:

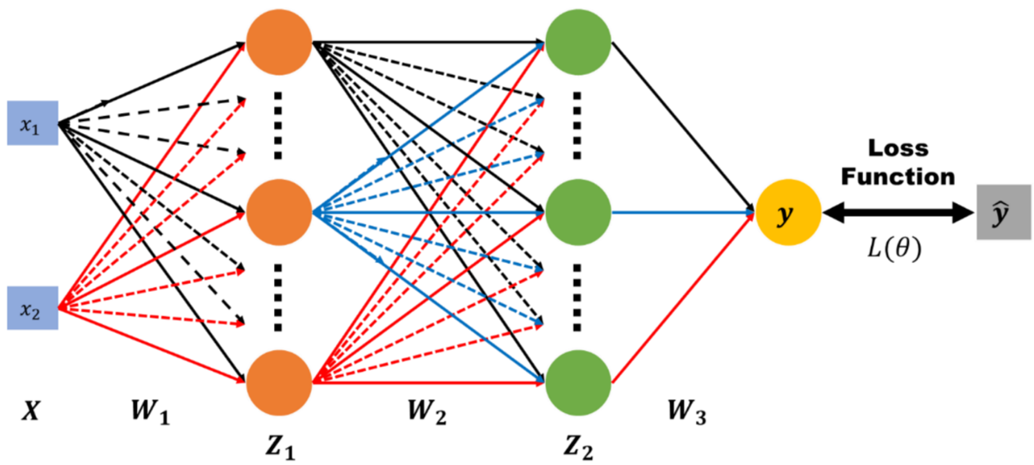
在這次 LAB 中，要達到以下事項：

(1) 只用 numpy 和其他標準程式庫架設一個 2 hidden layer 的 Deep network。

(2) 透過計算 Loss，使用反向傳播來更新 weight。

(3) 最後能讓預測準確度達到 90% 以上。

2 hidden layer deep network 架構如下：



1. $x1, x2$: *nerual network inputs*
2. X : $[x1, x2]$
3. y : *nerual network outputs* s
4. $\hat{y}$: *ground truth*
5. L(θ) : *loss function*
6. $W1, W2, W3$: *weight matrix of network layer*s
7. B1, B2, B3 : bias *matrix of network layer*s

整個 LAB 流程：

1. 初始化各個 layer 的節點數和 layer 之間的 weight, bias。
2. 正向傳播得到一個預測值 y_pred
3. 計算 y_pred 與 ground truth 之間的 loss
4. 使用反向傳播，計算 weight 對 loss 的影響
5. 透過前一步計算的 gradient 來更新新的 weight
6. 重複執行 2.-5.直到收斂

Experiment Setups:

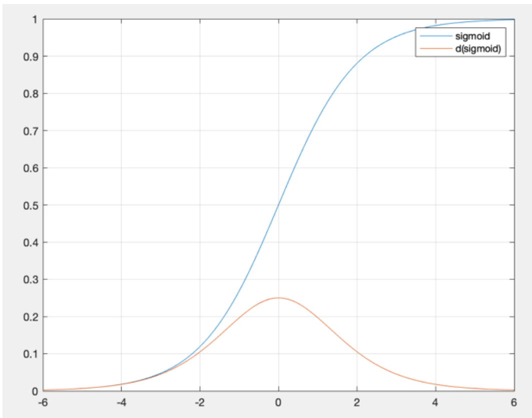
A. Sigmoid functions

Sigmoid function 因為 exp 的關係，可以擴大領先優勢，同時可以讓輸出限制在 [0,1] 之間，在 label 只有兩類時常常被使用在最後的輸出層。

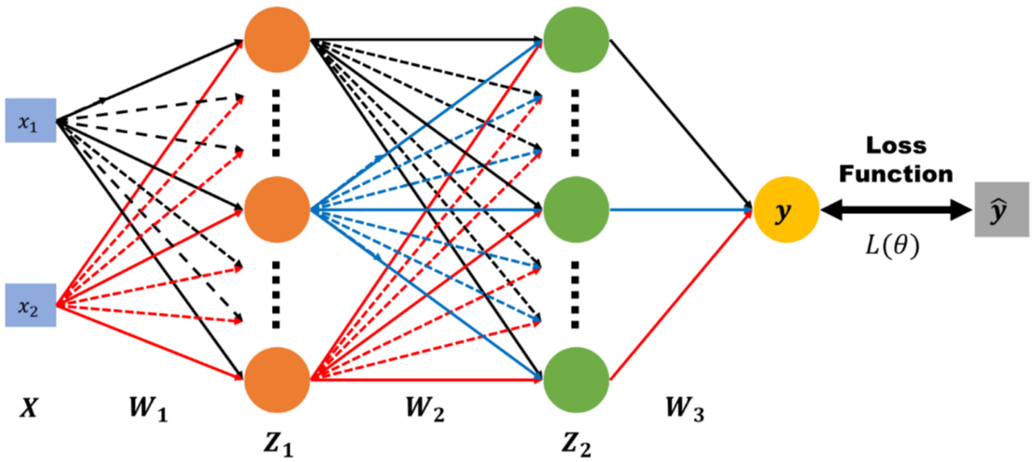
(在下一節呈現 Result 的部分，因為作業指定，所以都是以 sigmoid 函數作為 activation function)

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$[sigmoid(x)]' = sigmoid(x) * (1 - sigmoid(x))$$



B. Neural Network



同 Introduction 介紹，另外說明幾點：

W1, W2, W3, B1, B2, B3 在初始化時是隨機產生對應維度的矩陣。

Forward propagation: $Z_{i+1} = \sigma(W_{i+1}^{T}\, Z_i + B)$

Loss function: $L(\theta) = \frac{\sum_{i=1}^{n}(y-\hat{y})^2}{n}$ (mean square error)

C. Back propagation

∇W_3 , ∇B_3 :

$$\frac{\partial L}{\partial W_3^T} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial y'}\frac{\partial y'}{\partial W_3^T} = \sigma'(y) * 2(y-\hat{y}) \cdot Z_2^T = \nabla W_3^T$$

$$\frac{\partial L}{\partial B_3} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial y'} = \sigma'(y) * 2(y-\hat{y}) = \nabla B_3$$

∇W_2 , ∇B_2 :

$$\frac{\partial L}{\partial W_2^T} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial y'}\frac{\partial y'}{\partial Z_2}\frac{\partial Z_2}{\partial Z'}\frac{\partial Z_2'}{\partial W_2^T}$$

$$= \left[\sigma'(Z_2) * (W_3 \cdot (\sigma'(y) * 2(y-\hat{y})))\right] \cdot Z_2^T = \nabla W_2^T$$

$$\frac{\partial L}{\partial B_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial y'}\frac{\partial y'}{\partial Z_2}\frac{\partial Z_2}{\partial Z_2'} \cdot 1$$

$$= \left[\sigma'(Z_2) * (W_3 \cdot (\sigma'(y) * 2(y-\hat{y})))\right] = \nabla B_2$$

∇W_1 , ∇B_1

$$\frac{\partial L}{\partial W_1^T} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial y'}\frac{\partial y'}{\partial Z_2}\frac{\partial Z_2}{\partial Z_2'}\frac{\partial Z_2'}{\partial Z_1}\frac{\partial Z_1}{\partial Z_1'}\frac{\partial Z_1'}{\partial W_1^T}$$

$$= \left[\sigma'(Z_1) * W_2 \cdot \left[\sigma'(Z_2) * (W_3 \cdot \sigma'(y) * 2(y-\hat{y}))\right]\right] \cdot X^T = \nabla W_1^T$$

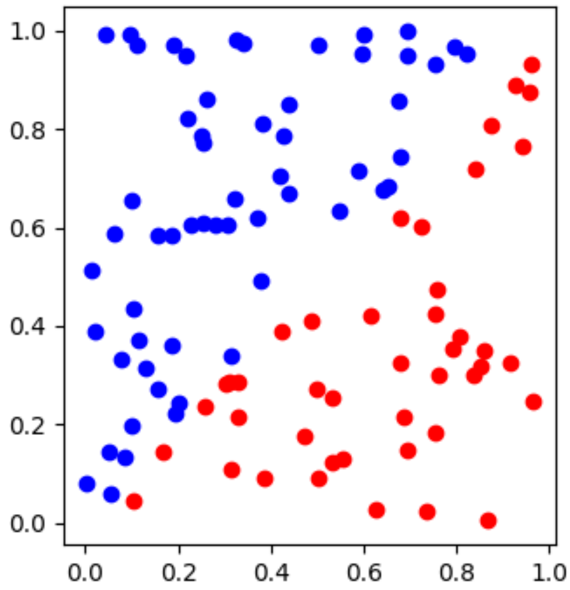
$$\frac{\partial L}{\partial B_1} = \sigma'(Z_1) * W_2 \left[\sigma'(Z_2) * (W_3 \cdot \sigma'(y) * 2(y-\hat{y}))\right] = \nabla B_1^T$$

" $*$ " is the element-wise operator

Result of Testing:

在 Lab 中使用了兩種 data 作測試：

(1) linear data:



訓練 linear data 使用的參數：

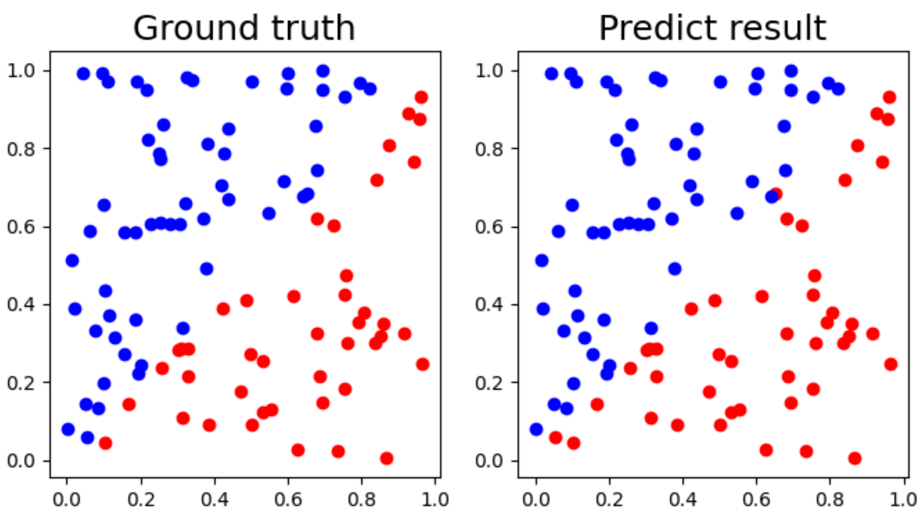
節點數：(inputLayer, hiddenLayer1, hiddenLayer2, outputLayer) = (2, 5, 5, 1)

Learning rate: 0.01

Activate function: sigmoid

Optimizer: SGD

結果比較：

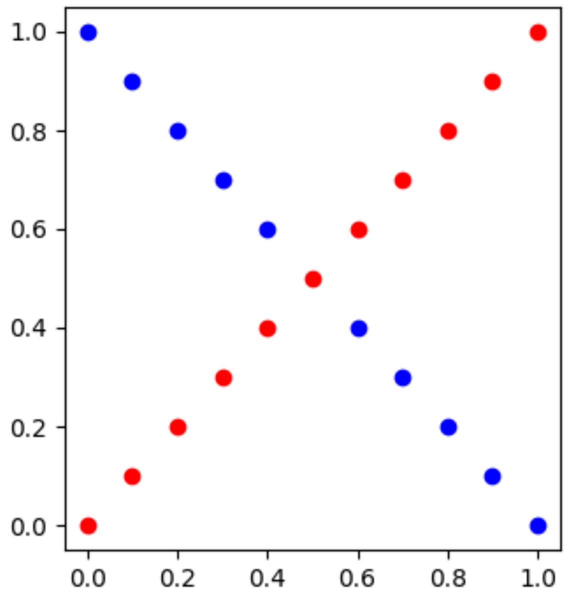


訓練時的 loss 跟 testing accuracy：

```
epoch:          0, loss:0.37650149, accuracy:57.00%
epoch:        500, loss:0.24830636, accuracy:56.00%
epoch:       1000, loss:0.24854323, accuracy:57.00%
epoch:       1500, loss:0.26020150, accuracy:46.00%
epoch:       2000, loss:0.25394168, accuracy:54.00%
epoch:       2500, loss:0.26126942, accuracy:49.00%
epoch:       3000, loss:0.24219162, accuracy:60.00%
epoch:       3500, loss:0.24876497, accuracy:54.00%
epoch:       4000, loss:0.05491345, accuracy:95.00%
epoch:       4500, loss:0.03915000, accuracy:92.00%
epoch:       5000, loss:0.03255377, accuracy:94.00%
epoch:       5500, loss:0.03823855, accuracy:93.00%
epoch:       6000, loss:0.03193779, accuracy:96.00%
epoch:       6500, loss:0.01194818, accuracy:98.00%
epoch:       7000, loss:0.01443738, accuracy:99.00%
epoch:       7500, loss:0.02092372, accuracy:97.00%
epoch:       8000, loss:0.01893835, accuracy:98.00%
epoch:       8500, loss:0.01063031, accuracy:99.00%
epoch:       9000, loss:0.01388598, accuracy:98.00%
epoch:       9500, loss:0.00769935, accuracy:99.00%
epoch:      10000, loss:0.02373350, accuracy:97.00%
epoch:      10500, loss:0.02222102, accuracy:97.00%
epoch:      11000, loss:0.01278821, accuracy:98.00%
epoch:      11500, loss:0.01637581, accuracy:98.00%
epoch:      12000, loss:0.00996847, accuracy:99.00%
epoch:      12500, loss:0.00086174, accuracy:100.00%
epoch:      13000, loss:0.00856620, accuracy:99.00%
epoch:      13500, loss:0.00266563, accuracy:100.00%
epoch:      14000, loss:0.00867435, accuracy:99.00%
epoch:      14500, loss:0.00565897, accuracy:99.00%
```

Learning Curve:

(2) non-linear data (XOR data):



<u>訓練 nonlinear data 使用的參數：</u>

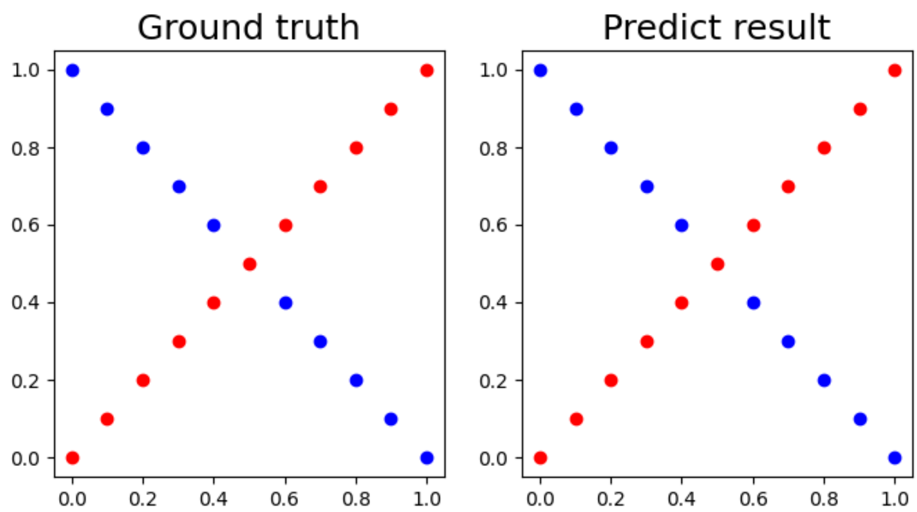
節點數：(inputLayer, hiddenLayer1, hiddenLayer2, outputLayer) = (2, 7, 7, 1)

Learning rate: 0.1

Activate function: sigmoid

Optimizer: SGD

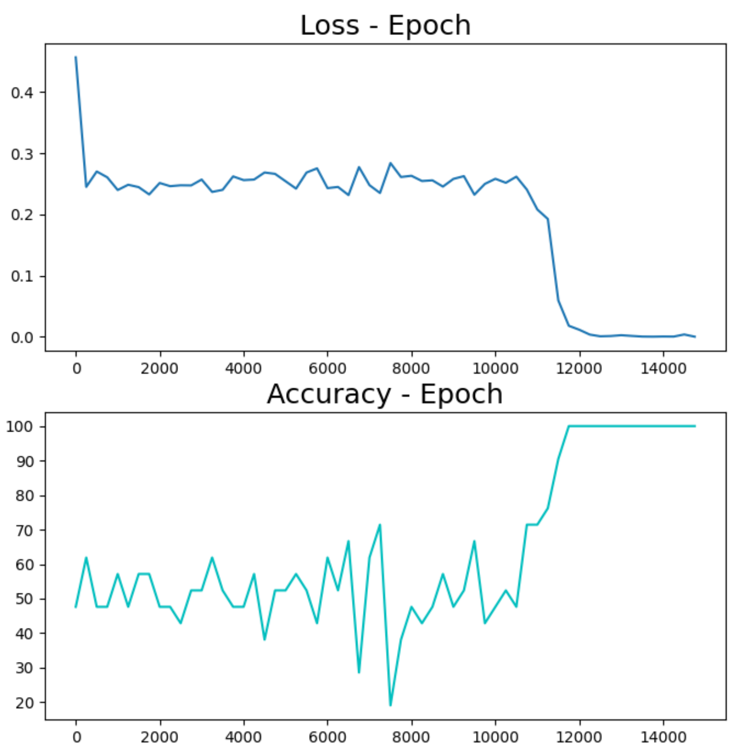
<u>結果比較：</u>



訓練時的 loss 跟 testing accuracy：

```
epoch:        0, loss:0.37650149, accuracy:57.00%
epoch:      500, loss:0.24830636, accuracy:56.00%
epoch:     1000, loss:0.24854323, accuracy:57.00%
epoch:     1500, loss:0.26020150, accuracy:46.00%
epoch:     2000, loss:0.25394168, accuracy:54.00%
epoch:     2500, loss:0.26126942, accuracy:49.00%
epoch:     3000, loss:0.24219162, accuracy:60.00%
epoch:     3500, loss:0.24876497, accuracy:54.00%
epoch:     4000, loss:0.05491345, accuracy:95.00%
epoch:     4500, loss:0.03915000, accuracy:92.00%
epoch:     5000, loss:0.03255377, accuracy:94.00%
epoch:     5500, loss:0.03823855, accuracy:93.00%
epoch:     6000, loss:0.03193779, accuracy:96.00%
epoch:     6500, loss:0.01194818, accuracy:98.00%
epoch:     7000, loss:0.01443738, accuracy:99.00%
epoch:     7500, loss:0.02092372, accuracy:97.00%
epoch:     8000, loss:0.01893835, accuracy:98.00%
epoch:     8500, loss:0.01063031, accuracy:99.00%
epoch:     9000, loss:0.01388598, accuracy:98.00%
epoch:     9500, loss:0.00769935, accuracy:99.00%
epoch:    10000, loss:0.02373350, accuracy:97.00%
epoch:    10500, loss:0.02222102, accuracy:97.00%
epoch:    11000, loss:0.01278821, accuracy:98.00%
epoch:    11500, loss:0.01637581, accuracy:98.00%
epoch:    12000, loss:0.00996847, accuracy:99.00%
epoch:    12500, loss:0.00086174, accuracy:100.00%
epoch:    13000, loss:0.00856620, accuracy:99.00%
epoch:    13500, loss:0.00266563, accuracy:100.00%
epoch:    14000, loss:0.00867435, accuracy:99.00%
epoch:    14500, loss:0.00565897, accuracy:99.00%
```

Learning Curve:



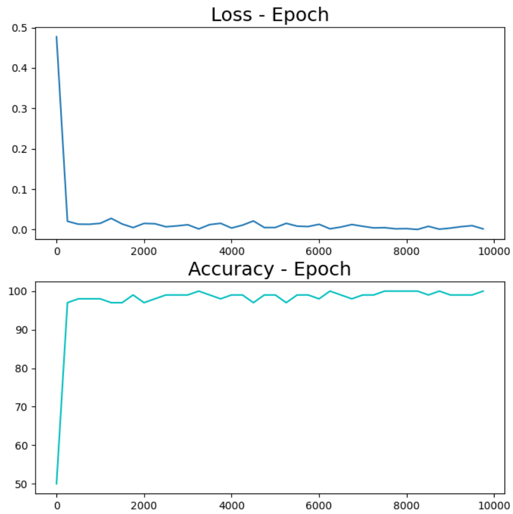
Discussion:

A. different learning rate

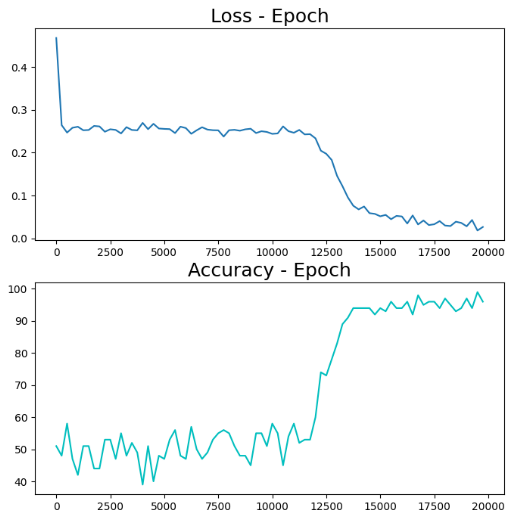
我分別嘗試以下各種 learning rate (其他參數無更改)。

在 linear 的 case 中

Learning rate = 0.1:



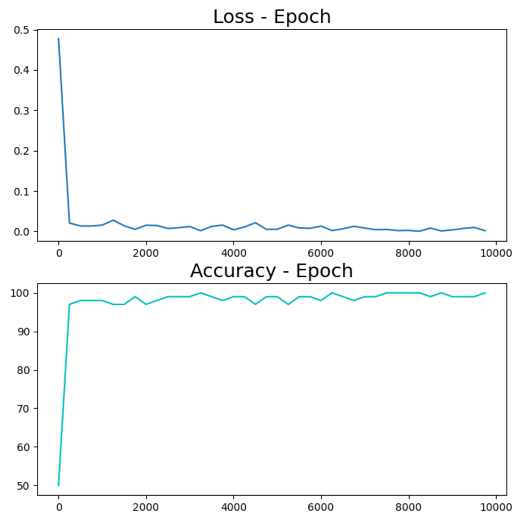
Learning rate = 0.001:’



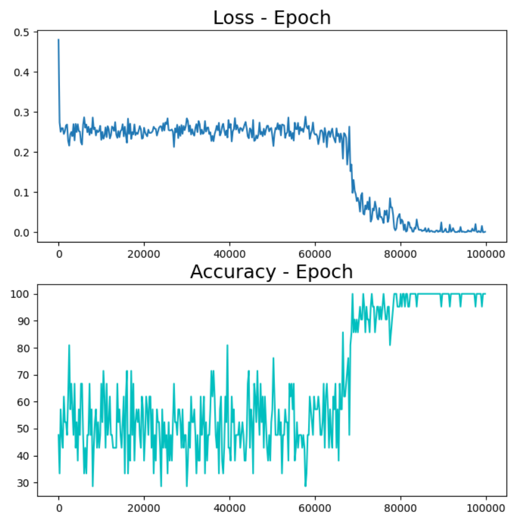
雖然都能收斂，但可以發現當 learning rate 太小時，所需要收斂的 epoch 就越多。

Non-linear case 的結果：

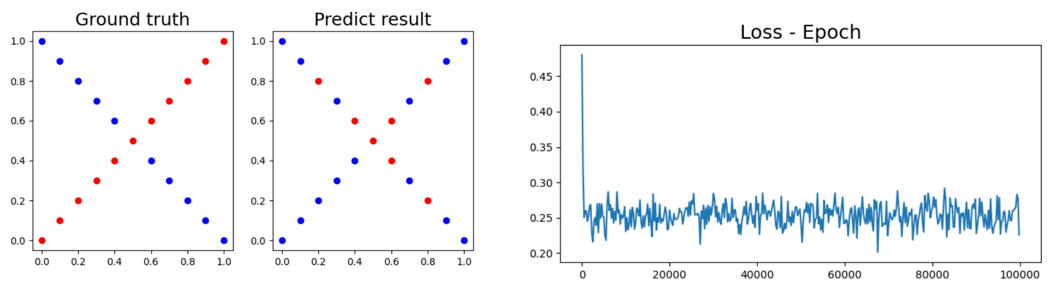
Learning rate = 1:



Learning rate = 0.01:



一樣有隨著 learning rate 減小而時間增加的趨勢，值得注意的是，當 learning rate= 10e^-3 時，是找不到解的(100000 epoch)，下圖所示。



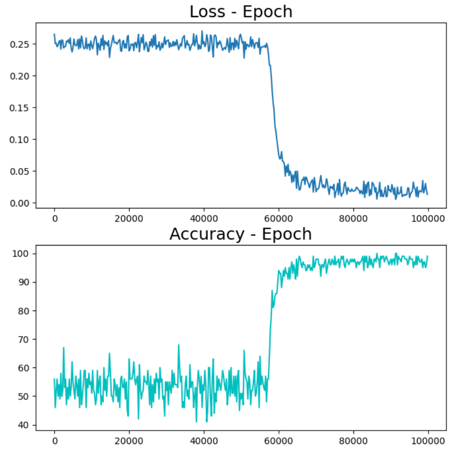
B. Different Hidden Units

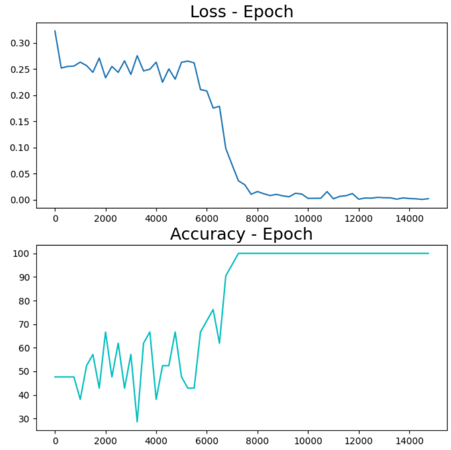
我先嘗試了極限的

節點數：(inputLayer, hiddenLayer1, hiddenLayer2, outputLayer) = (2, 2, 2, 1)

發現在 linear 跟 nonlinear 的 case 儘管時間花費較多，但都能找到不錯的模型

Linear: Nonlinear:

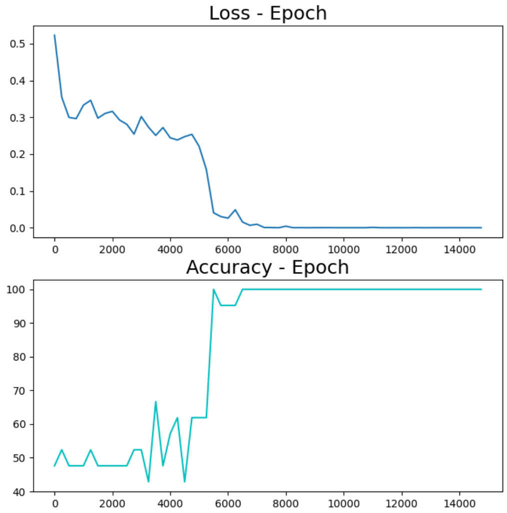




所以原本預期當 hidden units 越多，應該收斂越快，但在 nonlinear 的 case 裡面發現並沒有太大差別，也就是過多的 unit 可能只會徒增計算而已。

Nonlinear:

節點數：(inputLayer, hiddenLayer1, hiddenLayer2, outputLayer) = (2, 15, 15, 1)



C. Try without activation functions

不使用 activation function 時，因為非線性項不存在，所以預期只有 linear 可以 train，但實際在嘗試時，再計算 loss 常常會有 overflow 的狀況出現，所以連 linear 的狀況都無法使用，目前想到的方法是更改 loss function (e.g. Cross entropy)，但還沒有實現出來。

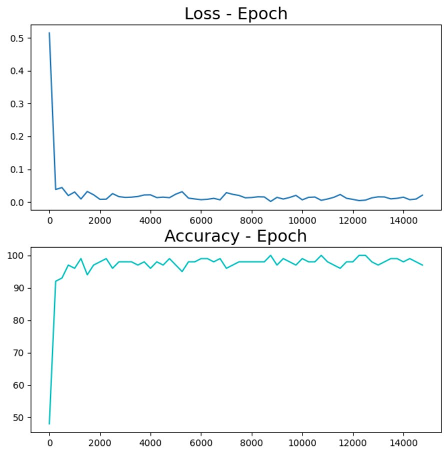
Extra:

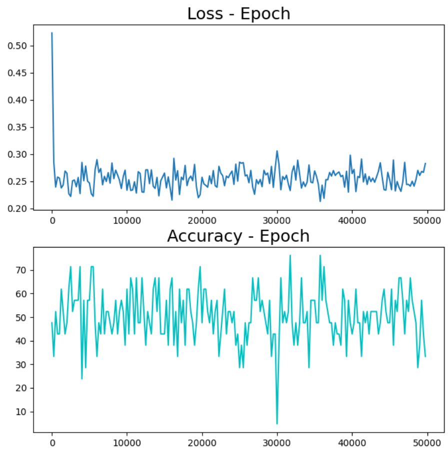
Implement different activation functions

承前面嘗試不使用 activation function，我先只在最後一層使用 sigmoid，其他一樣不加：

Linear activation functions:

Linear case: Non-Linear case



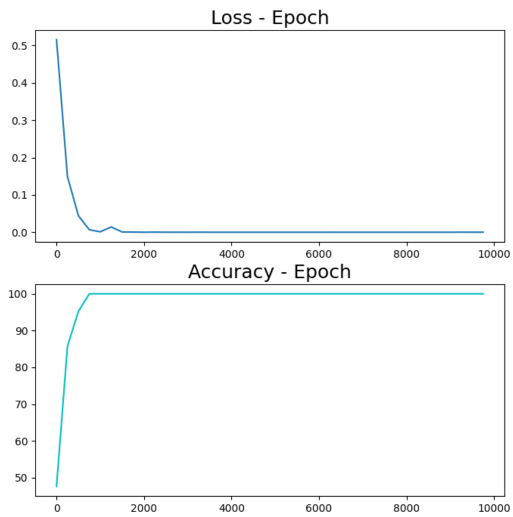


跟前面猜測相同，非線性的資料在不使用非線性 activation 下，是無法達成好的分類方法的。但我們可以看到，在線性資料裡面還是可以得到一個很高的預測準確率。

Tanh:

使用 tanh(x)當作 activation function:

以 nonlinear 資料為例：



發現 tanh 作為 activation function 在這個 lab 裡面可以更有效的讓資料分類。

ReLU:

雖然不太確定原因，但 ReLU 在 nonlinear 資料裡面並沒能得到成功分類效果，有試著嘗試 debug，但目前可能會嘗試改成複合式 ReLU(好幾個合在一起的)。