

DLP LAB1

1. Introduction

實作具有 2 層 hidden layer 的 Fully-Connected Neural Network 來將預測 input data 的分類，並藉由 Backpropagation 來加速 Gradient Descent 中計算 Gradient 步驟，其中實作可以任意修改層數功能。

2. Experiment setups

A. Sigmoid functions

為非線性方程式，利用它作為 activation function 解決非線性問題。

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

```
def sigmoid(x):  
    return 1.0 / (1.0 + np.exp(-x))  
  
def derivative_sigmoid(x):  
    """  
    Input: the value that output from sigmoid function.  
    """  
    return np.multiply(x, 1.0 - x)
```

上圖為 sigmoid 以及 derivative sigmoid function，sigmoid function 被用於 Forward Pass 的計算中，derivative sigmoid function 則用於 Backward Pass 的計算，其中需特別注意的是，derivative sigmoid function 的 input 為經過 sigmoid function 的 value。

B. Neural network

```
Network = NeuralNetwork(layer_dims=[2, 10, 10, 1], epoch=5000, lr=0.1, print_step=500)
```

2 層 Hidden layer 內皆有 10 個 hidden units · Learning rate 設為 0.1 · Loss function 使用 Cross Entropy ·

$$L_{Cross\ Entropy} = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

```
def compute_loss(self, predict, label):  
    """  
    Args:  
        predict: (1, data_num) ndarray  
        label: (data_num, 1) ndarray  
    Using Cross entropy as loss function.  
    """  
    m = label.shape[0]  
    ce = (1./m) * (-np.dot(label.T, np.log(predict+self.__epsilon).T) - np.dot((1-label).T, np.log(1-predict+self.__epsilon).T))  
    return ce
```

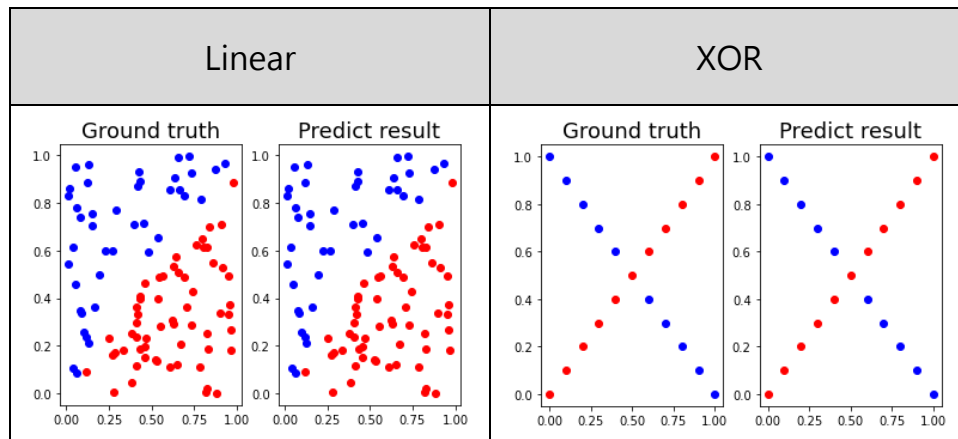
當 Loss 小於 0.001 時停止 · Epoch 上限為 100000 。

C. Backpropagation

- i. 初始化 Neural Network 的所有 Weight
- ii. 由 input layer 往 output layer 做 forward pass · 計算出所有 neuron 的 output
- iii. 再由 Neural Network 的 output 與實際 label 計算出 loss (誤差)
- iv. loss 由 output layer 往 input layer 做 backward pass (相當於一個反向的 Neural Network) · 並計算出每個 weight 對 loss 的偏微 (即該 neuron 對誤差的影響)
- v. 利用 weight 對 loss 的偏微去更新 weight
- vi. 重複步驟 ii.~v.直到 loss 夠小

3. Results of your testing

A. Screenshot and comparison figure



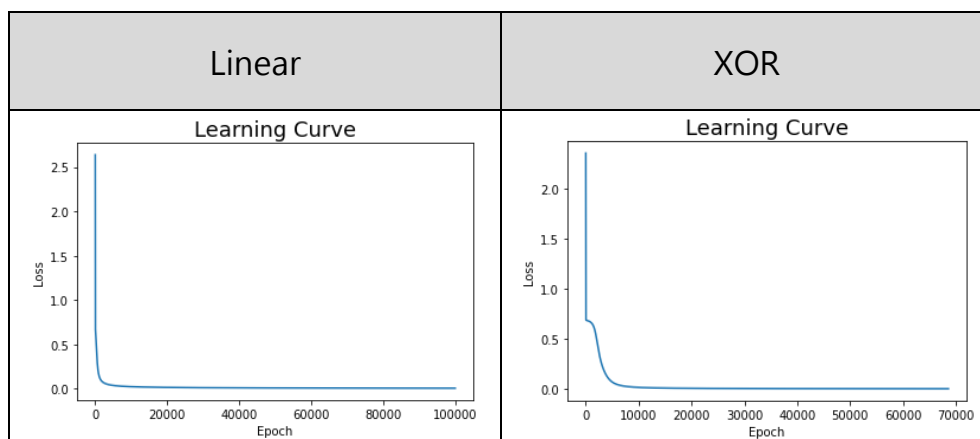
可以看到兩種 data 的預測皆是準確的。

B. Show the accuracy of your prediction

Linear	XOR
Test result: Accuracy: 1.0 Wrong Prediction Count: 0	Test result: Accuracy: 1.0 Wrong Prediction Count: 0

兩種 data 的準確率皆為 100%

C. Learning curve



<pre>---Start training--- Epoch: 5000 Loss: 0.037731177020179744 Accuracy: 0.999 Epoch: 10000 Loss: 0.023623442967184348 Accuracy: 0.999 Epoch: 15000 Loss: 0.018347718358493074 Accuracy: 1.0 Epoch: 20000 Loss: 0.015449065470767825 Accuracy: 1.0 Epoch: 25000 Loss: 0.013563340884476643 Accuracy: 1.0 Epoch: 30000 Loss: 0.01221267553913938 Accuracy: 1.0 Epoch: 35000 Loss: 0.011183431173799989 Accuracy: 0.999 Epoch: 40000 Loss: 0.010364693133339938 Accuracy: 0.999 Epoch: 45000 Loss: 0.009692658581952585 Accuracy: 0.999 Epoch: 50000 Loss: 0.009127759458873041 Accuracy: 0.999 Epoch: 55000 Loss: 0.008644001060918253 Accuracy: 0.999 Epoch: 60000 Loss: 0.00822350817095752 Accuracy: 0.999 Epoch: 65000 Loss: 0.007853524957975061 Accuracy: 0.999 Epoch: 70000 Loss: 0.007524665476775386 Accuracy: 0.999 Epoch: 75000 Loss: 0.007229843254562294 Accuracy: 0.999 Epoch: 80000 Loss: 0.006963589450435454 Accuracy: 0.999 Epoch: 85000 Loss: 0.006721603371930527 Accuracy: 0.999 Epoch: 90000 Loss: 0.006500447265070504 Accuracy: 0.999 Epoch: 95000 Loss: 0.006297333650705982 Accuracy: 0.999 Epoch: 100000 Loss: 0.006109973740365605 Accuracy: 0.999 ---Training finished---</pre>	<pre>---Start training--- Epoch: 5000 Loss: 0.06749298994824526 Accuracy: 1.0 Epoch: 10000 Loss: 0.014225098652219515 Accuracy: 1.0 Epoch: 15000 Loss: 0.007344676849219058 Accuracy: 1.0 Epoch: 20000 Loss: 0.004853871112569819 Accuracy: 1.0 Epoch: 25000 Loss: 0.00358819711186702 Accuracy: 1.0 Epoch: 30000 Loss: 0.002827155244875424 Accuracy: 1.0 Epoch: 35000 Loss: 0.002320904448788362 Accuracy: 1.0 Epoch: 40000 Loss: 0.001960643854580286 Accuracy: 1.0 Epoch: 45000 Loss: 0.0016915968382205805 Accuracy: 1.0 Epoch: 50000 Loss: 0.0014832511913688063 Accuracy: 1.0 Epoch: 55000 Loss: 0.0013172912802666744 Accuracy: 1.0 Epoch: 60000 Loss: 0.0011820705355488708 Accuracy: 1.0 Epoch: 65000 Loss: 0.0010698358410713882 Accuracy: 1.0 ---Training finished---</pre>
---	---

由上表可以看到 Linear data 較快收斂，而 XOR data 則是先維持在差不多的 loss 之後，才逐漸收斂。

D. Anything you want to present

Linear	XOR
Test Prediction: [[3.13274303e-08] [1.58410070e-09] [9.99999994e-01] [2.43348744e-09] [1.30383592e-08] [1.45990663e-07] [1.00000000e+00] [9.99999999e-01] [1.52700869e-09] [1.00000000e+00] [9.99560231e-01] [2.07563617e-09] [2.97048508e-09] [2.43040277e-09] [3.73636718e-09] [9.99999998e-01] [6.11596858e-05] [1.51282650e-09] [1.00000000e+00] [9.99999999e-01] [9.99999669e-01] [4.53082216e-09]]	Test Prediction: [[3.32566931e-04] [9.99501473e-01] [6.44744607e-04] [9.99518339e-01] [1.12448025e-03] [9.99513880e-01] [1.60007709e-03] [9.99403368e-01] [1.81012014e-03] [9.96494371e-01] [1.67797238e-03] [1.34715636e-03] [9.95192192e-01] [9.91442485e-04] [9.99279114e-01] [7.00859996e-04] [9.99619542e-01] [4.92220706e-04] [9.99720047e-01] [3.51019396e-04] [9.99754573e-01]]

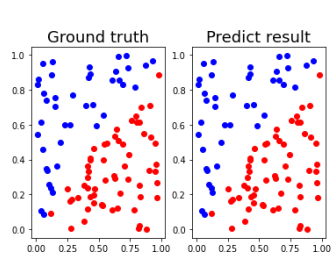
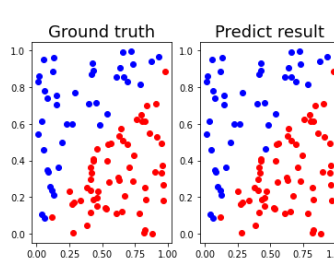
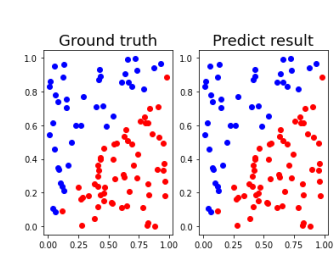
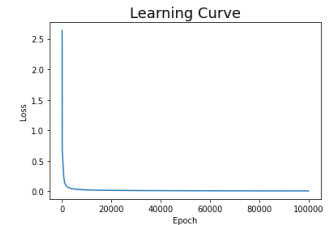
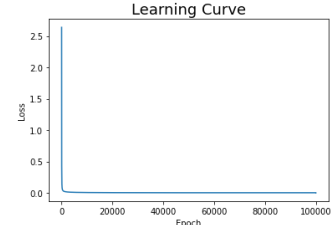
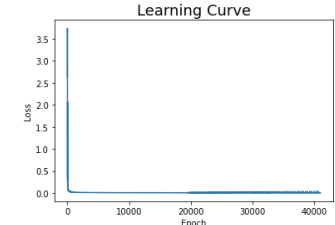
由上表可以看到 Network 對於兩種 data 的輸出值，Linear data 的輸出值都非常接近 0 或是 1，而 XOR data 的輸出值相對來說比較沒那麼接近 0 或 1。

4. Discussion

A. Try different learning rates

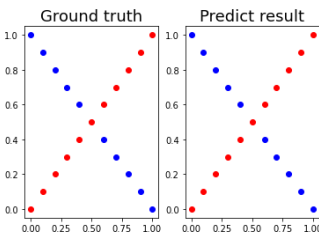
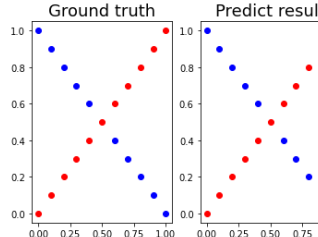
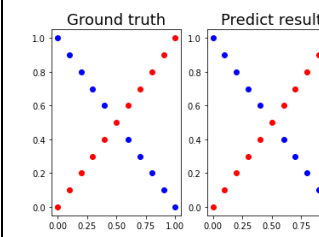
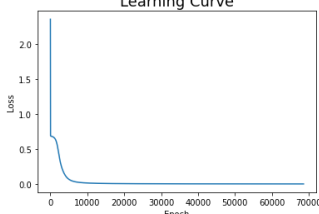
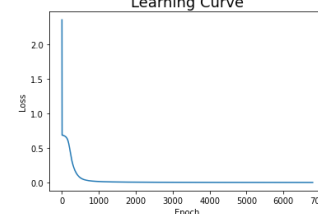
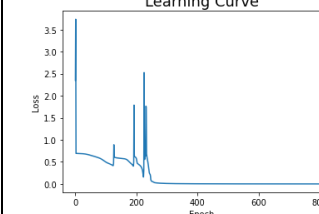
固定兩層 Hidden unit 皆為 10。

i. Linear

0.1	1	10
		
		
Test result: Accuracy: 1.0 Wrong Prediction Count: 0	Test result: Accuracy: 1.0 Wrong Prediction Count: 0	Test result: Accuracy: 1.0 Wrong Prediction Count: 0

由上表可以看到雖然結果都是預測正確，但是在 learning curve 的部分是有些差異，設為 1 時比較快達到較小的 loss 值，設為 10 時則更快收斂，且有時 loss 會往上升，因此 Learning curve 看起來有上下震盪的感覺。

ii. XOR

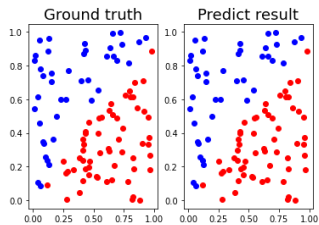
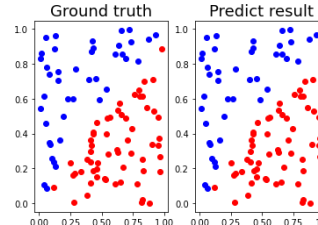
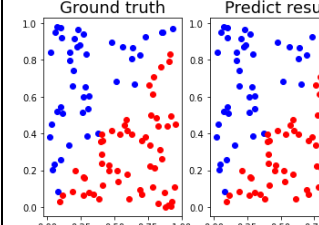
0.1	1	10
		
		
Test result: Accuracy: 1.0 Wrong Prediction Count: 0	Test result: Accuracy: 1.0 Wrong Prediction Count: 0	Test result: Accuracy: 1.0 Wrong Prediction Count: 0

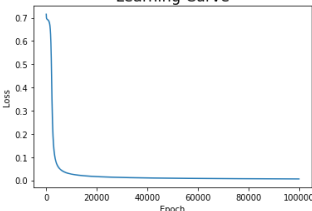
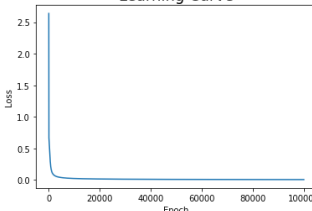
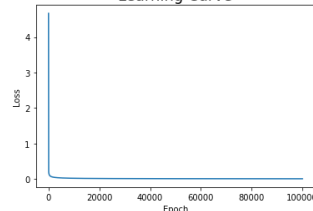
同樣結果也是都預測正確，但是相較於 linear data，更明顯看到設為 1 很快收斂，而設為 10 則又更快收斂，並且過程中 loss 有大幅度的上升，因此 learning curve 看起來震盪很大。

B. Try different numbers of hidden units

固定 learning rate 為 0.1。

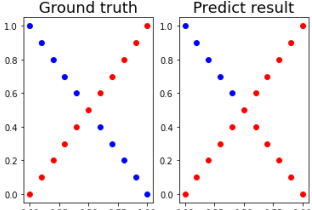
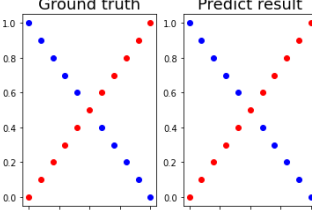
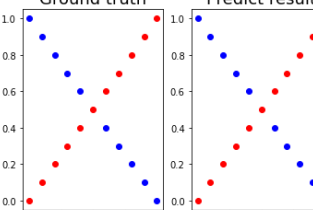
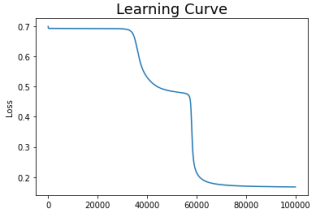
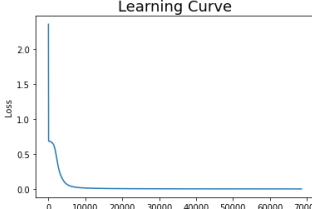
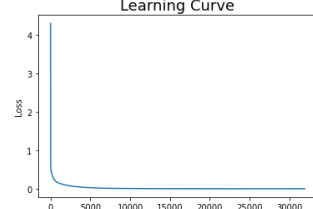
i. Linear

(2, 2)	(10, 10)	(100, 100)
		

		
<p>Test result: Accuracy: 1.0 Wrong Prediction Count: 0</p>	<p>Test result: Accuracy: 1.0 Wrong Prediction Count: 0</p>	<p>Test result: Accuracy: 1.0 Wrong Prediction Count: 0</p>

結果都預測正確，設為(2, 2)時初始的 Loss 最小，再來是(10, 10)，而(100, 100)則是最大的，不過最後都是收斂到差不多的 loss。

ii. XOR

(2, 2)	(10, 10)	(100, 100)
		
		
<p>Test result: Accuracy: 0.7619047619047619 Wrong Prediction Count: 5</p>	<p>Test result: Accuracy: 1.0 Wrong Prediction Count: 0</p>	<p>Test result: Accuracy: 1.0 Wrong Prediction Count: 0</p>

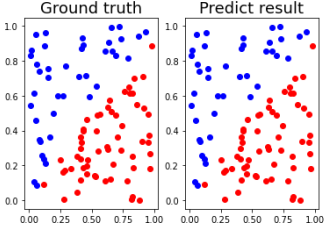
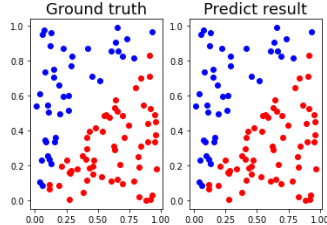
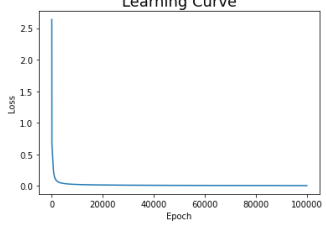
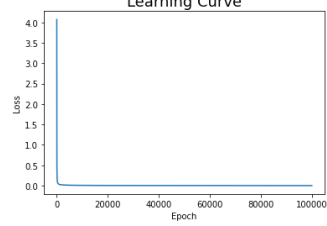
設為(2, 2)的結果讓準確率下降了約 24%，loss 下降較為緩慢，並且數值也較高，

而設為(100, 100)時的 learning curve 更快達到收斂，並且不像(10, 10)有先維持一下才往下降。

C. Try without activation functions

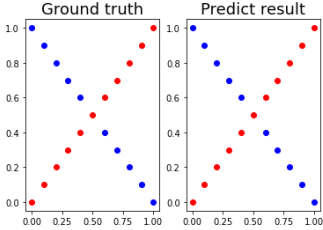
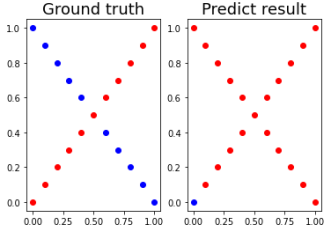
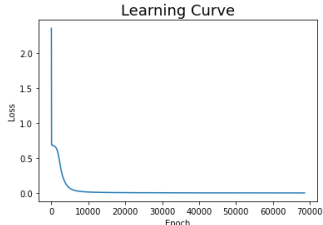
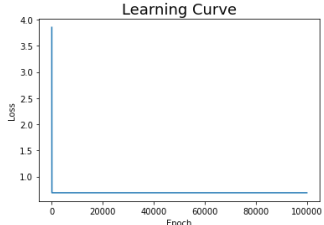
固定兩層 Hidden unit 皆為 10，learning rate 為 0.1，without activation function 只有在 output layer 使用 sigmoid function。

i. Linear

With activation function	Without activation function
	
	
<p>Test result:</p> <p>Accuracy: 1.0</p> <p>Wrong Prediction Count: 0</p>	<p>Test result:</p> <p>Accuracy: 1.0</p> <p>Wrong Prediction Count: 0</p>

除了剛開始 Loss 值較大外，沒有 activation function 對於 Linear data 似乎是沒有甚麼很大的影響。

ii. XOR

With activation function	Without activation function
	
	
<p>Test result:</p> <p>Accuracy: 1.0</p> <p>Wrong Prediction Count: 0</p>	<p>Test result:</p> <p>Accuracy: 0.47619047619047616</p> <p>Wrong Prediction Count: 11</p>

不同於 Linear data，沒有 activation function 對於 XOR data 則是有非常大的影響，準確率直接下降了約 53%，並且在很前面的 epoch 開始 loss 就一直維持在很大的值，不再繼續下降。

D. Anything you want to share

修改 learning rate 後，計算 Loss function 時會 overflow，之後將 Network 加一個很小的數值丟進去計算就解決了。

5. Extra

- Implement different optimizers
- Implement different activation functions
- Implement convolutional layers