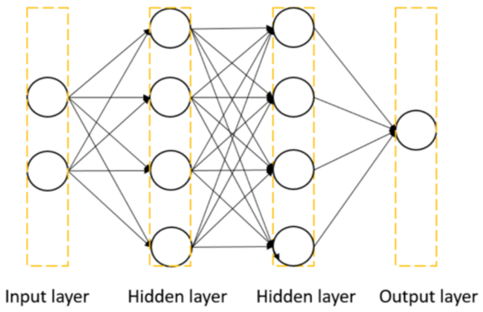
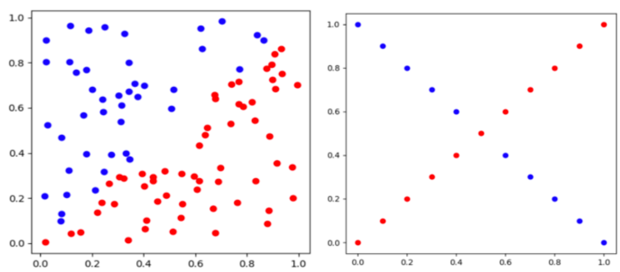
DLP Lab1 309551067 吳子涵

1. Introduction

在這次的 lab 中，需要實作一個有兩層 hidden layer 的 neural network，input layer 有 2 個 neuron，hidden layer 每層有 4 個 neuron，output layer 有 1 個 neuron，結構如下圖所示。



實驗分成兩部分，第一部分的 input 為下圖左，第二部分的 input 為下圖右。希望經由 neural network 的學習來分辨該資料點是紅點還是藍點。

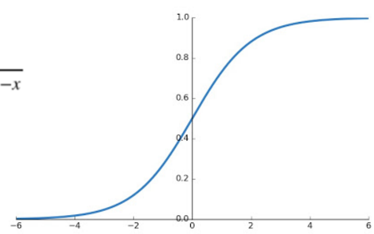


2. Experimental setups

A. Sigmoid function

Sigmoid function 會將 input 轉成介於 0 到 1 的值，且有嚴格遞增以及處處可微的特性，因此在神經網絡中是一個常用的 activation function。定義如下圖所示。

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

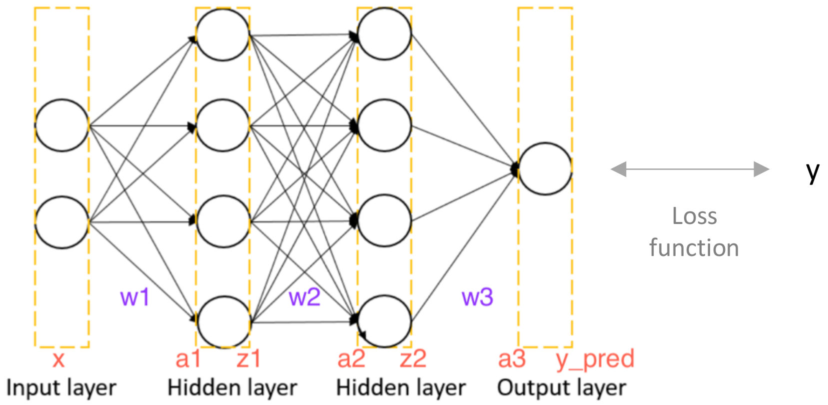


不過，sigmoid function 存在一些缺點：指數運算導致計算量大、易出現 gradient vanishing 的問題。

因為在 backward propagation 時需要 sigmoid function 的微分，以下是其推導。

$$
\begin{aligned}
\sigma(x)' &= \frac{d}{dx}(1+e^{-x})^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
$$

B. Neural Network



x : input data

wi : 第 i 層的參數

ai : 前一層 neuron 與 wi 內積

zi : ai 經過 sigmoid function 的結果

最終將 a3 經過 sigmoid function 後，即可得到 predict 的結果，再與 ground truth 計算 Loss function。

維度：(第一部分 n=100, 第二部分 n=21)

input layer: n*2

w1: 2*4

a1, z1, a2, z2: n*4

w2: 4*4

w3: 4*1

a3, y_pred: n*1

C. Backpropagation

Backpropagation 是一種常用來訓練 neural network 的方法，常與 Gradient Descent 結合使用。此方法利用 prediction 和 ground truth 的誤差來決定該如何更新權重。

Backpropagation 有兩個步驟：forward pass 和 backward pass。

(step1) forward pass

透過將同一層的每個 neuron 做 weighted sum，再經過 activation function，來更新下一層的參數，將數值從 input 一層層傳遞到 output。

$$a_1 = x \cdot w_1 \rightarrow z_1 = \mathrm{sigmoid}(a_1)$$
$$\rightarrow a_2 = z_1 \cdot w_2 \rightarrow z_2 = \mathrm{sigmoid}(a_2)$$
$$\rightarrow a_3 = z_2 \cdot w_3 \rightarrow y_p = \mathrm{sigmoid}(a_3)$$

```python
def forward(x, w1, w2, w3):
    a1 = np.dot(x, w1)
    z1 = sigmoid(a1)
    a2 = np.dot(z1, w2)
    z2 = sigmoid(a2)
    a3 = np.dot(z2, w3)
    y_pred = sigmoid(a3)
    return a1, z1, a2, z2, a3, y_pred
```

(step2) backward pass

利用 Gradient Descent 的方式，計算誤差對於每個權重的變化 ∂J/∂wi，來得知該如何更新參數，推導如下所示。

cost function = $J = -y \times \log(y_p) - (1-y) \times \log(1-y_p)$

$$\frac{dJ}{dw_3} = \underbrace{\frac{dJ}{dy_p}\frac{dy_p}{da_3}}_{\text{sigma3}}\frac{da_3}{dw_3} = \left(-y \times \frac{1}{y_p} - (1-y) \times \frac{-1}{1-y_p}\right) \times \left(y_p(1-y_p)\right) \times (z_2)$$
$$= \underline{(y_p - y)} \times z_2$$

$$\rightarrow w_3 = w_3 - \eta \frac{dJ}{dw_3}$$

$$\frac{dJ}{dw_2} = \frac{dJ}{dy_p}\frac{dy_p}{da_3}\frac{da_3}{dz_2}\frac{dz_2}{da_2}\frac{da_2}{dw_2} = \underbrace{(\text{sigma3}) \times (w_3) \times (z_2(1-z_2))}_{\text{sigma2}} \times z_1$$

$$\rightarrow w_2 = w_2 - \eta \frac{dJ}{dw_2}$$

$$\frac{dJ}{dw_1} = \frac{dJ}{dy_p}\frac{dy_p}{da_3}\frac{da_3}{dz_2}\frac{dz_2}{da_2}\frac{da_2}{dz_1}\frac{dz_1}{da_1}\frac{da_1}{dw_1} = (\text{sigma2}) \times (w_2) \times (z_1(1-z_1)) \times x$$

$$\rightarrow w_1 = w_1 - \eta \frac{dJ}{dw_1}$$

```python
def backward(x, y, y_pred, w1, w2, w3, z1, z2):
    sigma3 = y_pred - y
    w3 -= lr * np.dot(z2.T, sigma3)

    sigma2 = np.dot(sigma3, w3.T) * derivative_sigmoid(z2)
    w2 -= lr * np.dot(z1.T, sigma2)

    sigma1 = np.dot(sigma2, w2.T) * derivative_sigmoid(z1)
    w1 -= lr * np.dot(x.T, sigma1)

    return w1, w2, w3
```

3. Results of your testing

A. Screenshot and comparison figure

第一部分：

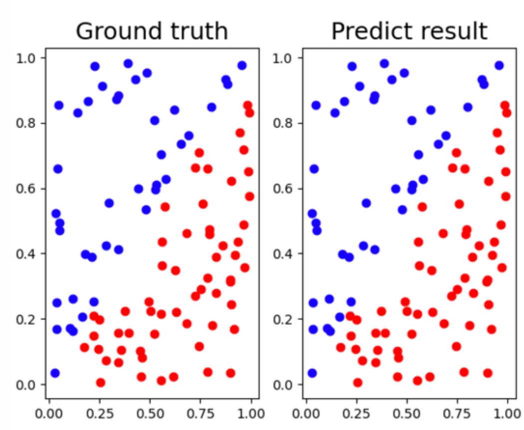
```
Training:
epoch 0 loss : 0.240018
epoch 1000 loss : 0.000076
epoch 2000 loss : 0.000015
epoch 3000 loss : 0.000006
epoch 4000 loss : 0.000003
epoch 5000 loss : 0.000002
epoch 6000 loss : 0.000001
epoch 7000 loss : 0.000001
epoch 8000 loss : 0.000001
epoch 9000 loss : 0.000000
epoch 10000 loss : 0.000000
```

```
Testing:
Prediction:
[[0.9999988 ]
 [0.00000039]
 [0.00000118]
 [0.00000055]
 [0.99999574]
 [0.00000064]
 [0.99999989]
 [0.99999987]
 [0.00000055]
 [0.99999989]
 [0.00000052]
 [0.99999876]
 [0.9999999 ]
 [0.00000036]
 [0.99999988]
 [0.00000104]
 [0.99999989]
 [0.99999985]
 [0.9997536 ]
 [0.00000055]
 [0.00000093]
 [0.99999989]
 [0.9984791 ]
```

```
 [0.00000046]
 [0.00462126]
 [0.99999989]
 [0.00000046]
 [0.00000051]
 [0.99999988]
 [0.99999985]
 [0.99998898]
 [0.00000038]
 [0.00000067]
 [0.99999989]
 [0.99999605]
 [0.00020648]
 [0.99999989]
 [0.00000076]
 [0.99999986]
 [0.99999989]
 [0.00000038]
 [0.99999989]
 [0.99999989]
 [0.9999999 ]
 [0.0000008 ]
 [0.0000006 ]
```

```
 [0.99999988]
 [0.99999989]
 [0.00000044]
 [0.99999989]
 [0.00000058]
 [0.00032692]
 [0.00000101]
 [0.99999989]
 [0.00000039]
 [0.99999987]
 [0.99999988]
 [0.99999984]
 [0.00000046]
 [0.00000082]
 [0.99999987]
 [0.99999989]
 [0.00000046]
 [0.00000051]
 [0.99999989]
 [0.00050721]
 [0.00000037]
 [0.99999984]
 [0.99677089]
 [0.00000042]
 [0.0000996 ]
```

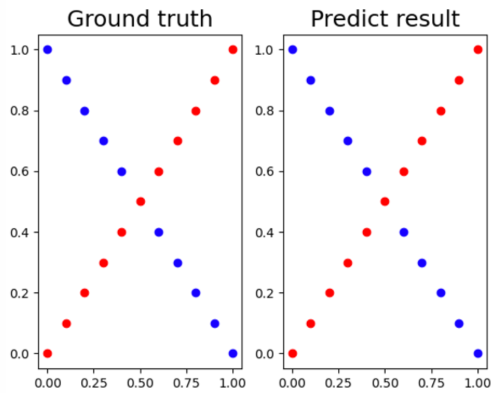
```
 [0.00000133]
 [0.99999989]
 [0.00000055]
 [0.00000039]
 [0.99999989]
 [0.99990035]
 [0.99997769]
 [0.00000059]
 [0.00000051]
 [0.99999985]
 [0.99982608]
 [0.99999988]
 [0.99999988]
 [0.99999989]
 [0.99999988]
 [0.00000618]
 [0.00000054]
 [0.00000056]
 [0.99996028]
 [0.0000004 ]
 [0.00000061]
 [0.99997399]
 [0.00000053]
 [0.99999984]
 [0.99999989]
 [0.00000084]
 [0.00000044]
 [0.00000051]
 [0.99999988]]
Accuracy: 100.00%
```



第二部分：

```
Training:
epoch 0 loss : 0.248979
epoch 1000 loss : 0.010420
epoch 2000 loss : 0.000126
epoch 3000 loss : 0.000022
epoch 4000 loss : 0.000008
epoch 5000 loss : 0.000004
epoch 6000 loss : 0.000002
epoch 7000 loss : 0.000001
epoch 8000 loss : 0.000001
epoch 9000 loss : 0.000001
epoch 10000 loss : 0.000001
```

```
Testing:
Prediction:
[[0.00006779]
 [0.99998478]
 [0.00018933]
 [0.99998297]
 [0.00055089]
 [0.99998011]
 [0.0009036 ]
 [0.9999705 ]
 [0.00078709]
 [0.99785611]
 [0.00047848]
 [0.000263  ]
 [0.99800656]
 [0.00014984]
 [0.99998097]
 [0.00009264]
 [0.99998314]
 [0.00006246]
 [0.99997722]
 [0.00004551]
 [0.99996364]]
Accuracy: 100.00%
```



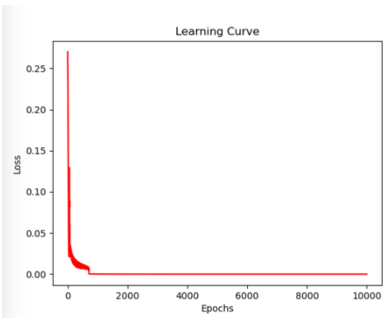
B. Show the accuracy of your prediction

第一部分：`Accuracy: 100.00%`

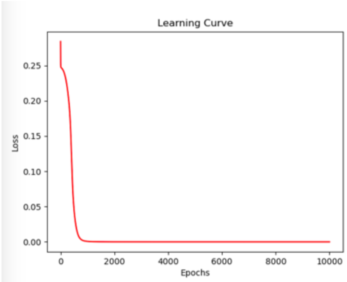
第二部分：`Accuracy: 100.00%`

C. Learning curve (loss, epoch curve)

第一部分：



第二部分：



D. Anything you want to present

因為這次 lab 的 data 較簡單，model 很快就能收斂，loss 很快就會很接近 0。而不論是第一部分的 data 或是第二部分的 data，準確率都能達到 100%，也就是所有紅點、藍點都能分對。

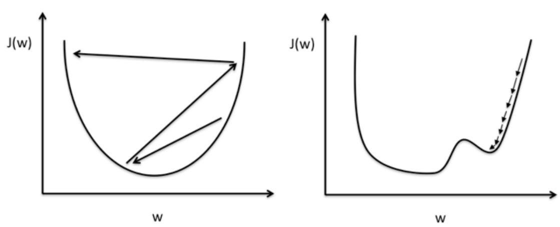
4. Discussion

A. Try different learning rates

learning rate 的選擇很重要：

若 learning rate 太大，雖然知道要往哪個方向走，但可能造成震盪的現象，導致無法收斂。(下圖左)

若 learning rate 太小，要花很多 iteration 才會收斂，且容易卡在 local minimum。(下圖右)



Large learning rate: Overshooting.

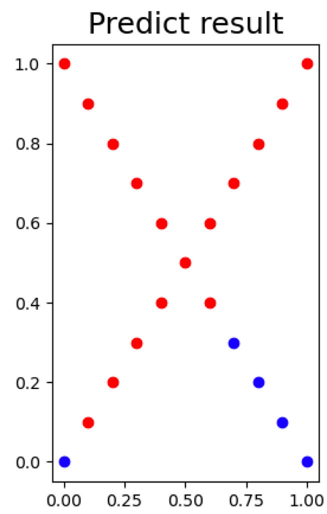
Small learning rate: Many iterations until convergence and trapping in local minima.

以第二部分的 data 為例

(1) learning rate = 10e-5

loss 會被困在 local minimum=0.24 左右，預測的結果不理想。

```
Training:
epoch 0 loss : 0.252767
epoch 5000 loss : 0.250997
epoch 10000 loss : 0.249864
epoch 15000 loss : 0.249133
epoch 20000 loss : 0.248655
epoch 25000 loss : 0.248337
epoch 30000 loss : 0.248122
epoch 35000 loss : 0.247973
epoch 40000 loss : 0.247865
epoch 45000 loss : 0.247785
epoch 50000 loss : 0.247721
epoch 55000 loss : 0.247670
epoch 60000 loss : 0.247625
epoch 65000 loss : 0.247585
epoch 70000 loss : 0.247548
epoch 75000 loss : 0.247514
epoch 80000 loss : 0.247480
epoch 85000 loss : 0.247447
epoch 90000 loss : 0.247415
epoch 95000 loss : 0.247383
epoch 100000 loss : 0.247351
```

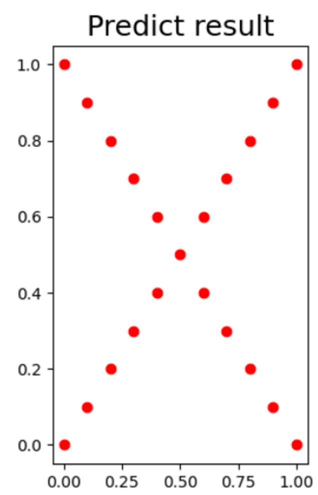


(2) learning rate = 0.05 (本次實驗所使用，結果如前面所示)

(3) learning rate = 2

loss 不斷跳動且不會收斂，預測的狀況很差。

```
Training:
epoch 0 loss : 0.271449
epoch 5000 loss : 0.523810
epoch 10000 loss : 0.523810
epoch 15000 loss : 0.476190
epoch 20000 loss : 0.523802
epoch 25000 loss : 0.357930
epoch 30000 loss : 0.367395
epoch 35000 loss : 0.523810
epoch 40000 loss : 0.523705
epoch 45000 loss : 0.523810
epoch 50000 loss : 0.476190
epoch 55000 loss : 0.476190
epoch 60000 loss : 0.523810
epoch 65000 loss : 0.476190
epoch 70000 loss : 0.476190
epoch 75000 loss : 0.518796
epoch 80000 loss : 0.476190
epoch 85000 loss : 0.476190
epoch 90000 loss : 0.476190
epoch 95000 loss : 0.476190
epoch 100000 loss : 0.523710
```



B. Try different numbers of hidden units

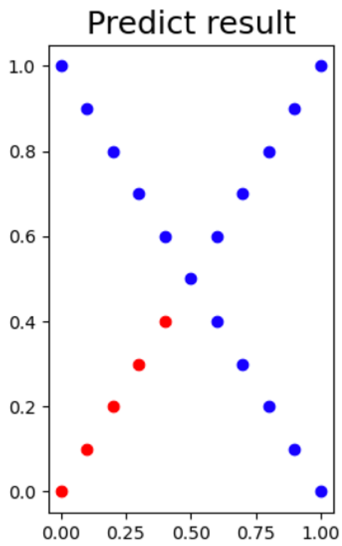
hidden unit 的數量會影響 model 的 capacity。

以第二部分的 data 為例

(1) hidden unit 數量=2

loss 會被困在 local minimum=0.19 左右，預測的結果不理想。

```
Training:
epoch 0 loss : 0.272856
epoch 5000 loss : 0.205280
epoch 10000 loss : 0.202417
epoch 15000 loss : 0.201142
epoch 20000 loss : 0.200336
epoch 25000 loss : 0.199758
epoch 30000 loss : 0.199315
epoch 35000 loss : 0.198959
epoch 40000 loss : 0.198664
epoch 45000 loss : 0.198415
epoch 50000 loss : 0.198200
epoch 55000 loss : 0.198012
epoch 60000 loss : 0.197845
epoch 65000 loss : 0.197695
epoch 70000 loss : 0.197560
epoch 75000 loss : 0.197438
epoch 80000 loss : 0.197325
epoch 85000 loss : 0.197222
epoch 90000 loss : 0.197126
epoch 95000 loss : 0.197037
epoch 100000 loss : 0.196955
```

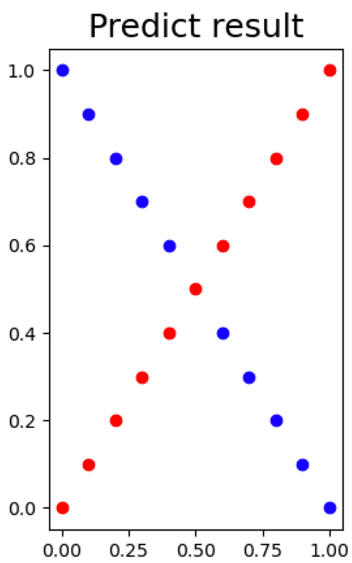


(2) hidden unit 數量=4 (本次實驗所使用，結果如前面所示)

(3) hidden unit 數量=10

loss 很快就能收斂，能準確預測結果。

```
Training:
epoch 0 loss : 0.429719
epoch 5000 loss : 0.000003
epoch 10000 loss : 0.000000
epoch 15000 loss : 0.000000
epoch 20000 loss : 0.000000
epoch 25000 loss : 0.000000
epoch 30000 loss : 0.000000
epoch 35000 loss : 0.000000
epoch 40000 loss : 0.000000
epoch 45000 loss : 0.000000
epoch 50000 loss : 0.000000
epoch 55000 loss : 0.000000
epoch 60000 loss : 0.000000
epoch 65000 loss : 0.000000
epoch 70000 loss : 0.000000
epoch 75000 loss : 0.000000
epoch 80000 loss : 0.000000
epoch 85000 loss : 0.000000
epoch 90000 loss : 0.000000
epoch 95000 loss : 0.000000
epoch 100000 loss : 0.000000
```

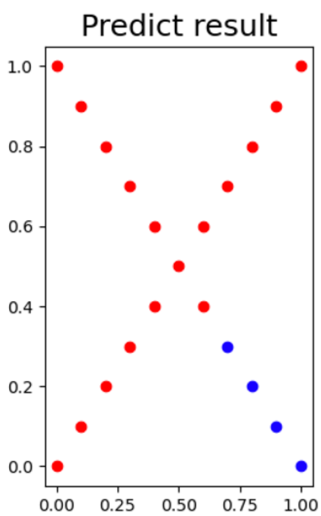


C. Try without sigmoid function

activation function 是訓練神經網絡中重要的一個角色，若沒有使用 activation function，那不管中間有幾層 hidden layer，神經網絡的 input 和 output 都會是線性關係，而導致無法產生好的預測結果。

以第二部分的 data 為例，neural network 中未加 sigmoid function 所產生的預測結果。

```
Training:
epoch 0 loss : 3.249709
epoch 5000 loss : 3.350134
epoch 10000 loss : 3.463471
epoch 15000 loss : 3.584110
epoch 20000 loss : 3.709632
epoch 25000 loss : 3.838643
epoch 30000 loss : 3.970202
epoch 35000 loss : 4.103606
epoch 40000 loss : 4.238288
epoch 45000 loss : 4.373760
epoch 50000 loss : 4.509592
epoch 55000 loss : 4.645384
epoch 60000 loss : 4.780765
epoch 65000 loss : 4.915385
epoch 70000 loss : 5.048915
epoch 75000 loss : 5.181049
epoch 80000 loss : 5.311510
epoch 85000 loss : 5.440058
epoch 90000 loss : 5.566498
epoch 95000 loss : 5.690689
epoch 100000 loss : 5.812548
```



D. Anything you want to share

透過這次的 lab，我也更了解 backpropagation 的計算原理以及物理意義。我將所有相關的算式都自己算了一次，再根據公式就能輕鬆得寫成程式。另外，經過我反覆驗證，我發現很多網路上分享的算式其實有一些推導細節是不正確的，雖然在簡單的 data 下還是會收斂，不過會收斂得比較慢。