Lab1-Backpropagation Report

313551104 黃暐洺

1. Introduction (20%)

本次實作了兩層 hidden layer 的神經網路，只使用 numpy 完成全部的運算，程式包含 layer 及 MLP 兩個 class，和其他數學運算的副程式，layer 可執行 feedforward 和 backpropagation，運用 linear 跟 XOR 兩個數據集做訓練與測試，最後比較各種參數之間在效能上的差異。

2. Experiment setups (30%) :

A. Sigmoid functions

```
def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))
```

B. Neural network

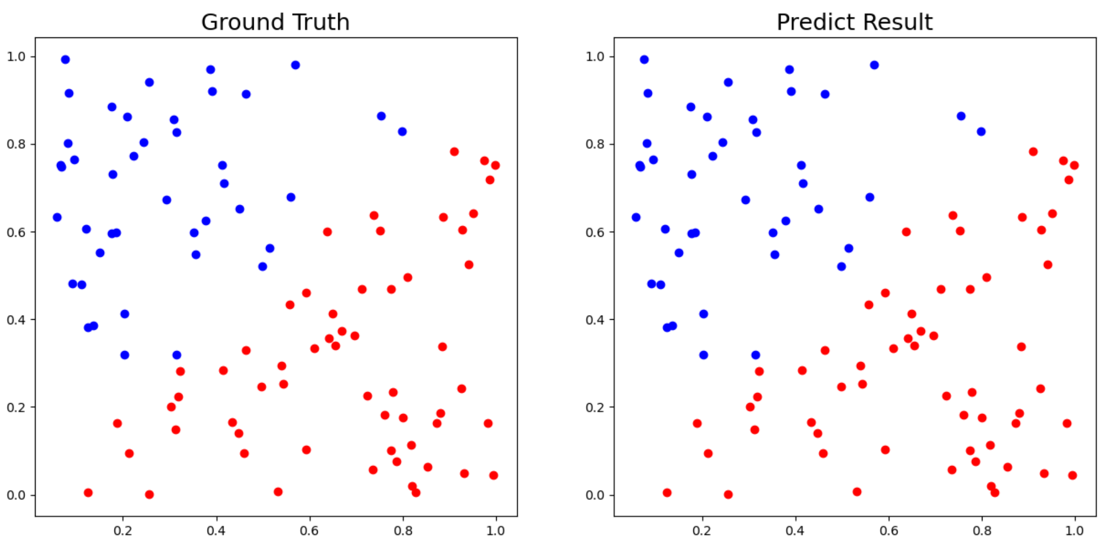
```
class MLP:
    def __init__(self, hidden_size = 5, learning_rate = 0.5, activate = "none"):
        self.learning_rate = learning_rate
        self.hidden1 = layer(2, hidden_size, activate)
        self.hidden2 = layer(hidden_size, hidden_size, activate)
        self.output = layer(hidden_size, 1, activate)
        self.loss = []

    def train(self, x, ground_truth, epoch = 100000):
        plt.figure(figsize = (30, 30))
        plt.subplot(2, 1, 1)
        plt.title('Learning Curves', fontsize = 30)
        plt.xlabel('epoch', fontsize = 30)
        plt.ylabel('loss', fontsize = 30)
        for i in range(epoch):
            self.hidden1.forward(x)
            self.hidden2.forward(self.hidden1.z)
            self.output.forward(self.hidden2.z)

            loss, gradient = self.cost(ground_truth)
            self.loss.append(loss)
            if not i % 5000:
                print("epoch ", i, " loss : ", loss)

            self.output.back(gradient, self.learning_rate)
            self.hidden2.back(self.output.gradient, self.learning_rate)
            self.hidden1.back(self.hidden2.gradient, self.learning_rate)
        plt.plot(self.loss)

        show_result(x, ground_truth, self.output.z)
        for i in range(ground_truth.size):
            print("Iter", i + 1," |    Ground truth: ", ground_truth[i], " |   prediction: ", self.output.z[i], "
        print("loss=", loss, " accuracy=", 100 * sum((self.output.z > 0.5) == (ground_truth == 1)) / ground_truth

    def cost(self, y_hat):
        return MSE(self.output.z, y_hat), y_grad(self.output.z, y_hat)

    def test(self, x, ground_truth):
        self.hidden1.forward(x)
        self.hidden2.forward(self.hidden1.z)
        self.output.forward(self.hidden2.z)
        loss, grad = self.cost(ground_truth)
        print("  ===== testing set =====  ")
        show_result(x, ground_truth, self.output.z)
        for i in range(ground_truth.size):
            print("Iter", i + 1," |    Ground truth: ", ground_truth[i], " |   prediction: ", self.output.z[i], "
        print("loss=", loss, " accuracy=", 100 * sum((self.output.z > 0.5) == (ground_truth == 1)) / ground_truth
```

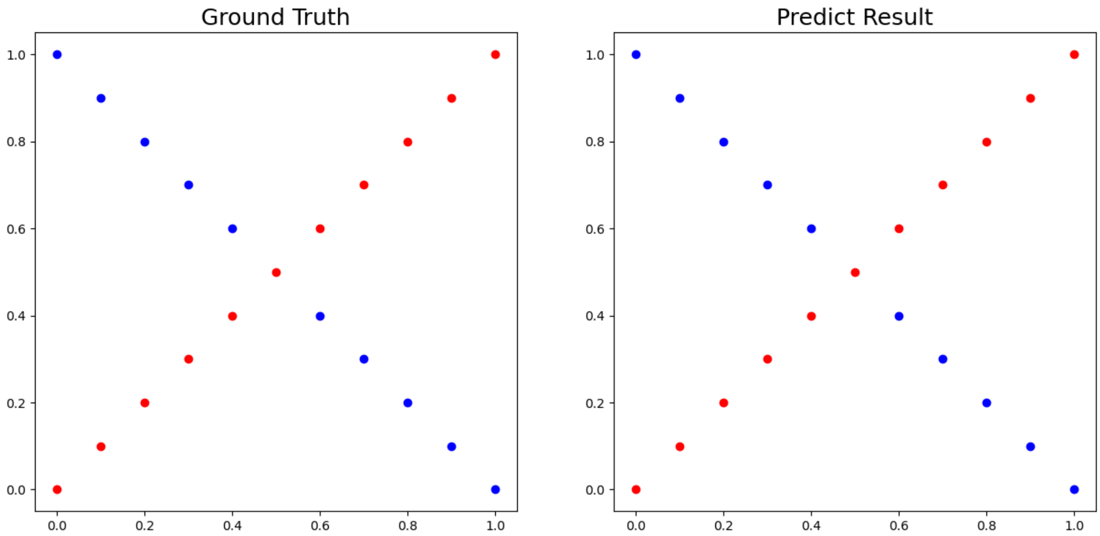
C. Backpropagation

```python
class layer:
    def __init__(self,input_size,output_size, activate = "none"):
        self.input_size = input_size
        self.output_size = output_size
        self.activate = activate
        self.gradient = 0
        self.w = np.random.normal(-1, 1, (input_size, output_size))
        self.b = np.random.normal(-1, 1, (1, output_size))
        self.a = []
        self.z = []

    def forward(self, x):
        self.x = x
        self.a = np.dot(x,self.w) + self.b
        if(self.activate == "sigmoid"):
            self.z = sigmoid(self.a)
        elif(self.activate == "ReLU"):
            self.z = ReLU(self.a)
        else:
            self.z = self.a

    def back(self, gradient, learning_rate):
        if self.activate == "sigmoid":
            gradient *= derivative_sigmoid(self.z)
        elif self.activate == "ReLU":
            gradient *= derivative_ReLU(self.z)
        else:
            pass

        self.w -= learning_rate * np.dot(self.x.T, gradient)
        self.b -= learning_rate * np.sum(gradient, axis = 0)
        self.gradient = np.dot(gradient, self.w.T)
```

3. Results of your testing (20%)

A. Screenshot and comparison figure





B. Show the accuracy of your prediction

Linear :

```
x1, y1 = generate_linear()
model = MLP(hidden_size = 5,activate = "sigmoid", learning_rate = 0.1)
model.train(x1, y1, epoch = 100000)
```

✓ 4.4s

```
epoch  0  loss :  0.3222392885432805
epoch  5000  loss :  0.02616356246458329
epoch  10000  loss :  0.01174779135501703
epoch  15000  loss :  0.008510572394455846
epoch  20000  loss :  0.00692084150529909
epoch  25000  loss :  0.005909300050339386
epoch  30000  loss :  0.00517039997620771
epoch  35000  loss :  0.004586251604458222
epoch  40000  loss :  0.004103229797072347
epoch  45000  loss :  0.003693583068616542
epoch  50000  loss :  0.003341050109749299
epoch  55000  loss :  0.003034907714084778
epoch  60000  loss :  0.002767363823042213
epoch  65000  loss :  0.0025323640557419116
epoch  70000  loss :  0.002325015394309402
epoch  75000  loss :  0.002141281055711663
epoch  80000  loss :  0.0019777979312595203
epoch  85000  loss :  0.001831751416627836
epoch  90000  loss :  0.0017007801040933124
epoch  95000  loss :  0.0015828986711753186
```

```
Iter 1  |   Ground truth:  [1]  |   prediction:  [0.9881353]  |
Iter 2  |   Ground truth:  [0]  |   prediction:  [4.49336206e-06]  |
Iter 3  |   Ground truth:  [1]  |   prediction:  [0.9999837]  |
Iter 4  |   Ground truth:  [1]  |   prediction:  [0.99998683]  |
Iter 5  |   Ground truth:  [0]  |   prediction:  [4.95955647e-06]  |
Iter 6  |   Ground truth:  [1]  |   prediction:  [0.99998707]  |
Iter 7  |   Ground truth:  [1]  |   prediction:  [0.99998807]  |
Iter 8  |   Ground truth:  [0]  |   prediction:  [0.00228932]  |
Iter 9  |   Ground truth:  [0]  |   prediction:  [0.0010142]  |
Iter 10  |   Ground truth:  [1]  |   prediction:  [0.99958826]  |
Iter 11  |   Ground truth:  [1]  |   prediction:  [0.99998858]  |
Iter 12  |   Ground truth:  [1]  |   prediction:  [0.93635657]  |
Iter 13  |   Ground truth:  [0]  |   prediction:  [5.21899493e-06]  |
Iter 14  |   Ground truth:  [1]  |   prediction:  [0.99996796]  |
Iter 15  |   Ground truth:  [1]  |   prediction:  [0.99998732]  |
Iter 16  |   Ground truth:  [1]  |   prediction:  [0.99998803]  |
Iter 17  |   Ground truth:  [0]  |   prediction:  [1.22098112e-05]  |
Iter 18  |   Ground truth:  [0]  |   prediction:  [4.76219678e-06]  |
Iter 19  |   Ground truth:  [0]  |   prediction:  [0.00043786]  |
Iter 20  |   Ground truth:  [0]  |   prediction:  [2.77989435e-05]  |
Iter 21  |   Ground truth:  [1]  |   prediction:  [0.99996743]  |
Iter 22  |   Ground truth:  [0]  |   prediction:  [4.49549571e-06]  |
Iter 23  |   Ground truth:  [0]  |   prediction:  [4.21171025e-06]  |
Iter 24  |   Ground truth:  [0]  |   prediction:  [2.60143534e-05]  |
Iter 25  |   Ground truth:  [0]  |   prediction:  [4.3213849e-06]  |
...
Iter 98  |   Ground truth:  [0]  |   prediction:  [1.16042308e-05]  |
Iter 99  |   Ground truth:  [0]  |   prediction:  [0.14894483]  |
Iter 100  |   Ground truth:  [1]  |   prediction:  [0.99949564]  |
loss= 0.0014764541109010209  accuracy= [100.] %
```

XOR :

```
x2, y2 = generate_XOR_easy()
model = MLP(hidden_size = 5,activate = "sigmoid", learning_rate = 0.1)
model.train(x2, y2, epoch = 100000)
```

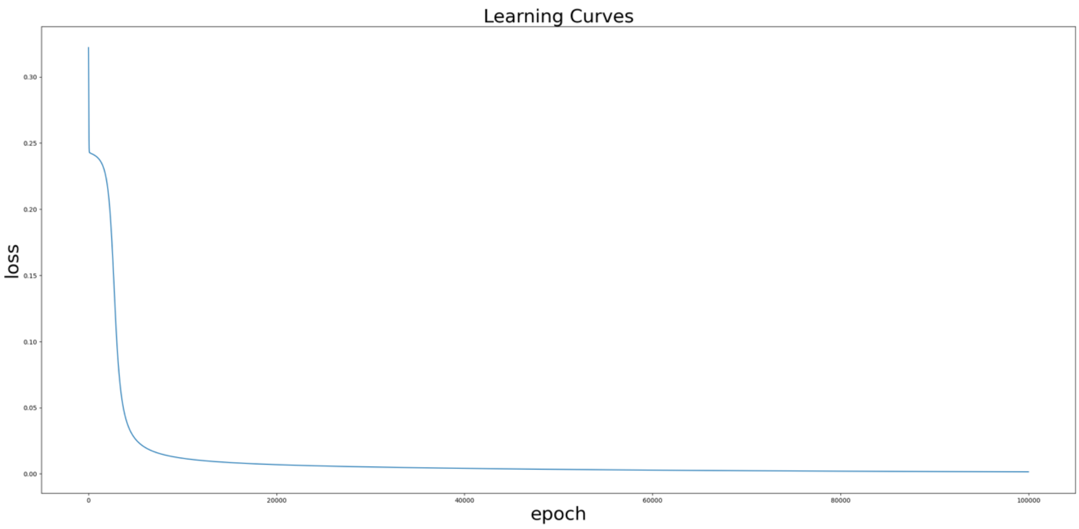
✓ 3.3s

```
epoch  0  loss :  0.28501257141830455
epoch  5000  loss :  0.2491668946915397
epoch  10000  loss :  0.24866809769428033
epoch  15000  loss :  0.24569798101133355
epoch  20000  loss :  0.2057230874273731
epoch  25000  loss :  0.05709604395989675
epoch  30000  loss :  0.01673334284997154
epoch  35000  loss :  0.005918474413453942
epoch  40000  loss :  0.00303345045423603
epoch  45000  loss :  0.0019160415536937942
epoch  50000  loss :  0.0013602939521317468
epoch  55000  loss :  0.0010379239924489946
epoch  60000  loss :  0.0008310455355576931
epoch  65000  loss :  0.0006885636392691007
epoch  70000  loss :  0.0005852067298738649
epoch  75000  loss :  0.0005072029378803439
epoch  80000  loss :  0.0004464710189918848
epoch  85000  loss :  0.0003979848040527405
epoch  90000  loss :  0.00035846803572880487
epoch  95000  loss :  0.0003257010223061783
```

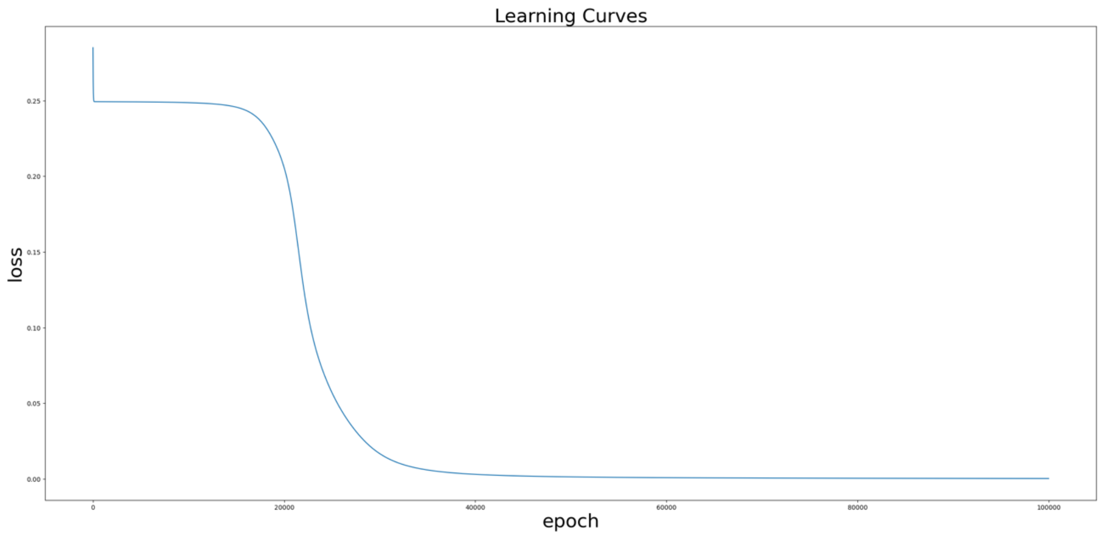
```
Iter 1  |    Ground truth:  [0]  |   prediction:  [0.01659925]  |
Iter 2  |    Ground truth:  [1]  |   prediction:  [0.99976639]  |
Iter 3  |    Ground truth:  [0]  |   prediction:  [0.01655754]  |
Iter 4  |    Ground truth:  [1]  |   prediction:  [0.99975685]  |
Iter 5  |    Ground truth:  [0]  |   prediction:  [0.01652328]  |
Iter 6  |    Ground truth:  [1]  |   prediction:  [0.99972007]  |
Iter 7  |    Ground truth:  [0]  |   prediction:  [0.01649688]  |
Iter 8  |    Ground truth:  [1]  |   prediction:  [0.99944487]  |
Iter 9  |    Ground truth:  [0]  |   prediction:  [0.01647875]  |
Iter 10  |    Ground truth:  [1]  |   prediction:  [0.9610959]  |
Iter 11  |    Ground truth:  [0]  |   prediction:  [0.0164692]  |
Iter 12  |    Ground truth:  [0]  |   prediction:  [0.01646849]  |
Iter 13  |    Ground truth:  [1]  |   prediction:  [0.9651592]  |
Iter 14  |    Ground truth:  [0]  |   prediction:  [0.01647683]  |
Iter 15  |    Ground truth:  [1]  |   prediction:  [0.98923106]  |
Iter 16  |    Ground truth:  [0]  |   prediction:  [0.01649432]  |
Iter 17  |    Ground truth:  [1]  |   prediction:  [0.98859891]  |
Iter 18  |    Ground truth:  [0]  |   prediction:  [0.01652105]  |
Iter 19  |    Ground truth:  [1]  |   prediction:  [0.98812384]  |
Iter 20  |    Ground truth:  [0]  |   prediction:  [0.01655702]  |
Iter 21  |    Ground truth:  [1]  |   prediction:  [0.98789783]  |
loss= 0.0002981355719449254  accuracy= [100.] %
```

C. Learning curve (loss, epoch curve)

Linear :



XOR:



D. Anything you want to present

Layer 初始化時參數的選擇很重要，挑得不好會直接造成 Loss 無法下降，或是在訓練初期 Loss 降低得慢，用常態分佈選出始值的模型，會比均勻分佈挑初始值得模型訓練的效果更好。

對於沒看過的資料，像 XOR 這種複雜的分佈無法得到很高的準確度，線性模型的 testing set 就算很大，準確率也很高。

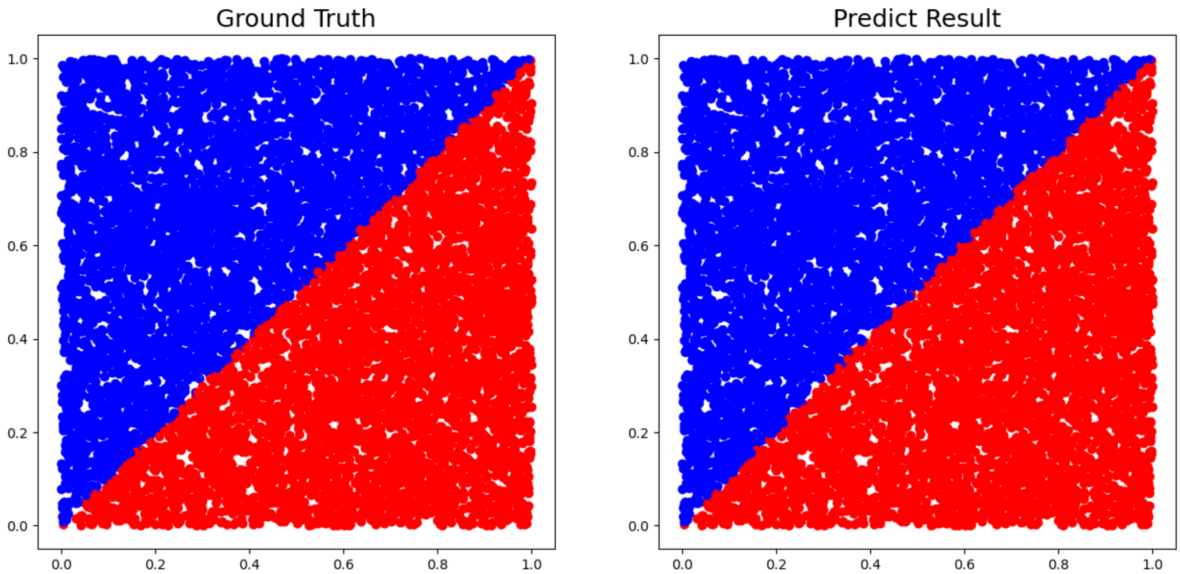
自行製作了產生更多點集的 generate_XOR_hard()，用於測試模型

```
def generate_XOR_hard(fraction = 0.1):
    inputs = []
    labels = []
    for i in range(int(1 / fraction) + 1):
        inputs.append([fraction * i, fraction * i])
        labels.append(0)
        if i == int(0.5 / fraction):
            continue
        inputs.append([fraction * i, 1 - fraction * i])
        labels.append(1)
    return np.array(inputs), np.array(labels).reshape(int(2 / fraction) + 1, 1)
```

以下是測試結果：

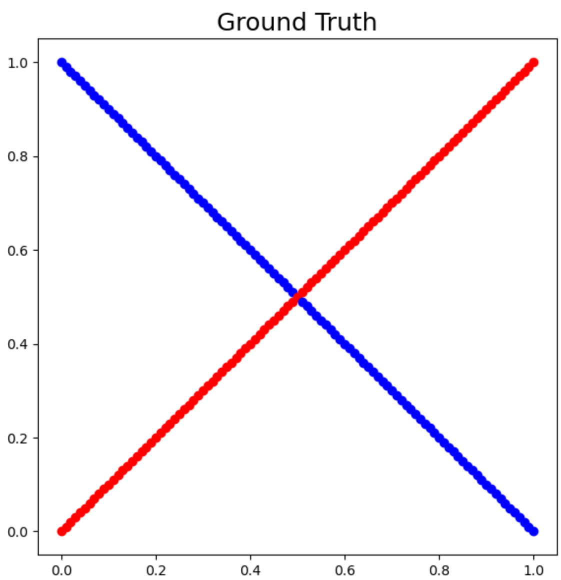
Linear：

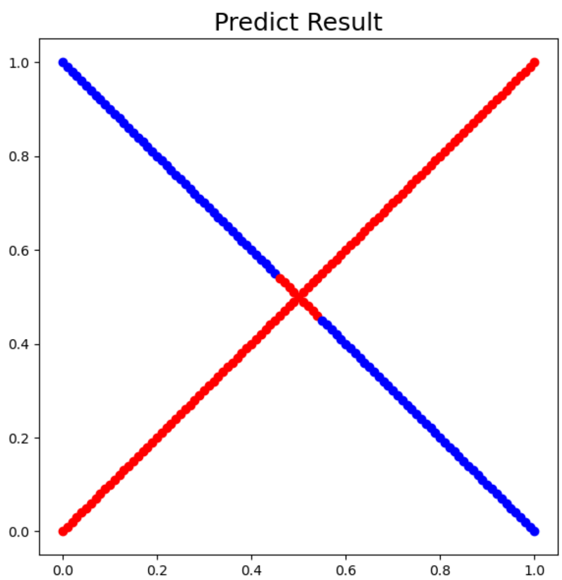
```
Iter 1  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 2  |    Ground truth:  [1]  |   prediction:  [0.99863226]  |
Iter 3  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 4  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 5  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 6  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 7  |    Ground truth:  [1]  |   prediction:  [1.00388766]  |
Iter 8  |    Ground truth:  [1]  |   prediction:  [0.99566165]  |
Iter 9  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 10  |    Ground truth:  [1]  |   prediction:  [1.00128008]  |
Iter 11  |    Ground truth:  [1]  |   prediction:  [0.99831033]  |
Iter 12  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 13  |    Ground truth:  [1]  |   prediction:  [0.99640321]  |
Iter 14  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 15  |    Ground truth:  [1]  |   prediction:  [1.00408681]  |
Iter 16  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 17  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 18  |    Ground truth:  [1]  |   prediction:  [0.99722167]  |
Iter 19  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 20  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 21  |    Ground truth:  [1]  |   prediction:  [0.99793719]  |
Iter 22  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 23  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 24  |    Ground truth:  [1]  |   prediction:  [0.99999761]  |
Iter 25  |    Ground truth:  [0]  |   prediction:  [0.]  |
...
Iter 9998  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 9999  |    Ground truth:  [1]  |   prediction:  [0.99671956]  |
Iter 10000  |    Ground truth:  [0]  |   prediction:  [0.]  |
loss= 0.004241871682561259  accuracy= [99.36] %
```



XOR :

```
Iter 1  |   Ground truth:  [0]  |   prediction:  [5.20694599e-14]  |
Iter 2  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 3  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 4  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 5  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 6  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 7  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 8  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 9  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 10  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 11  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 12  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 13  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 14  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 15  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 16  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 17  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 18  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 19  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 20  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 21  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 22  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 23  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 24  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 25  |   Ground truth:  [0]  |   prediction:  [0.]  |
...
Iter 199  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 200  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 201  |   Ground truth:  [1]  |   prediction:  [1.]  |
loss= 0.028381267564463724  accuracy= [96.0199005] %
```



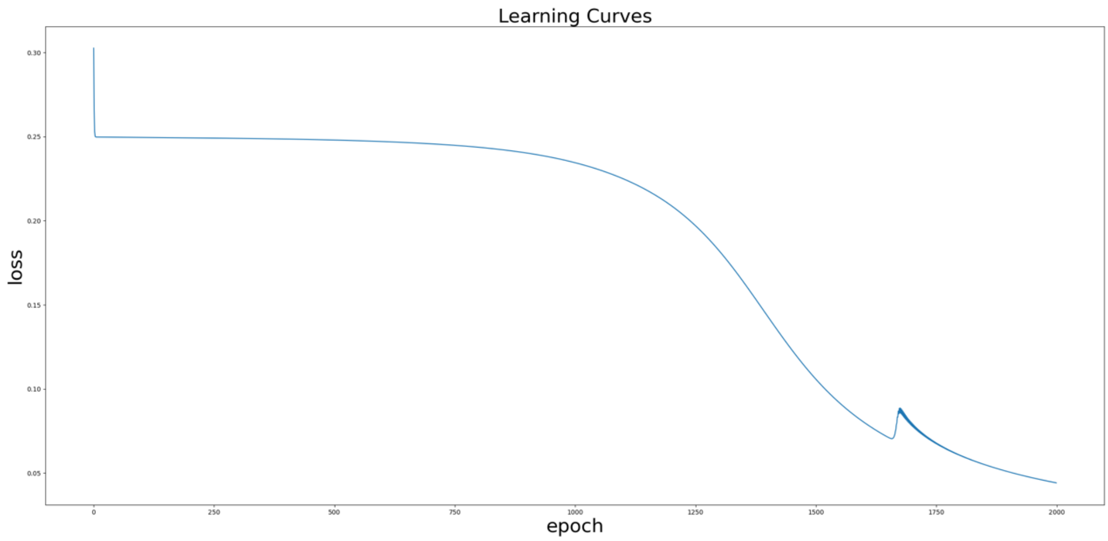


4. Discussion (30%)

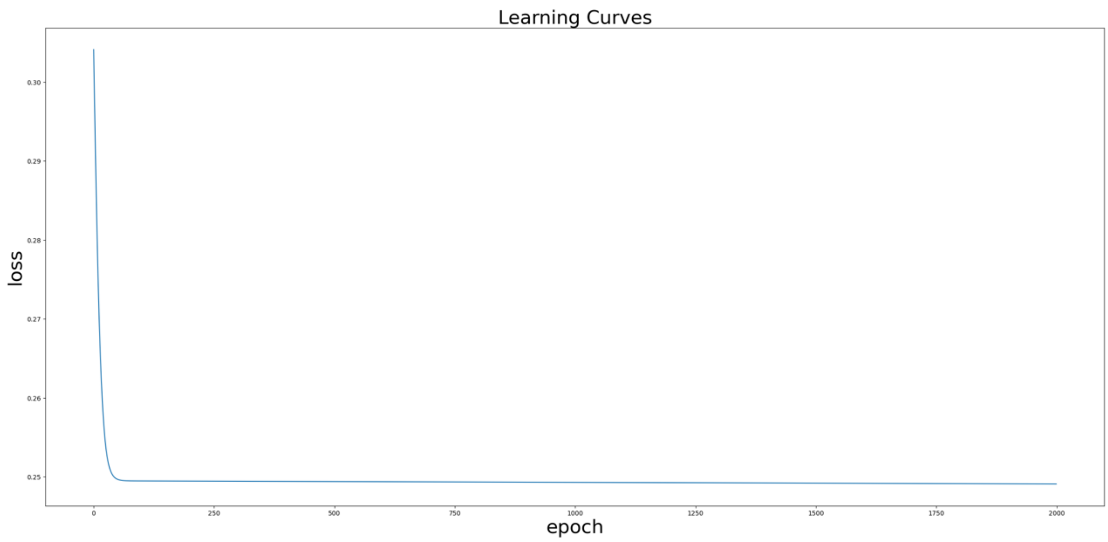
A. Try different learning rates

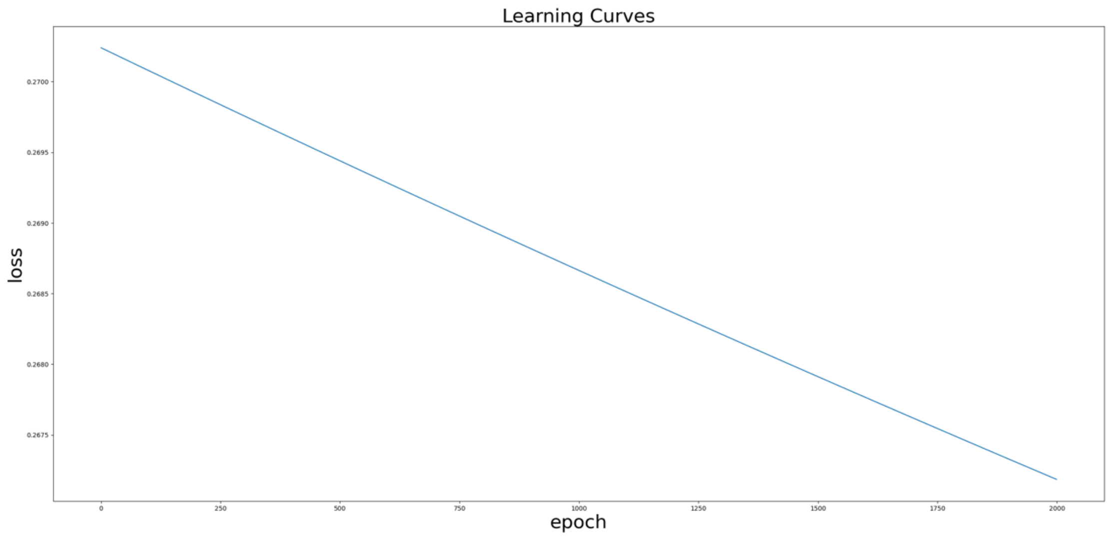
Learning rates 越小參數能進行越精確的調整，而 LR 越大能讓 Loss 收斂更快，若太大會修正過頭讓 Loss 不降反增、或是忽大忽小，但若 LR 太小的話，參數修正太少會讓訓練效果不顯著，需要更多迭代才能收斂。

LR 太大，Loss 不降反增，且前期 Loss 幾乎不變：



下面兩張圖可對比 LR 大小之差異，上方這張很陡峭(LR = 1)，下面的則是更接近一條直線(LR = 1e-3)：

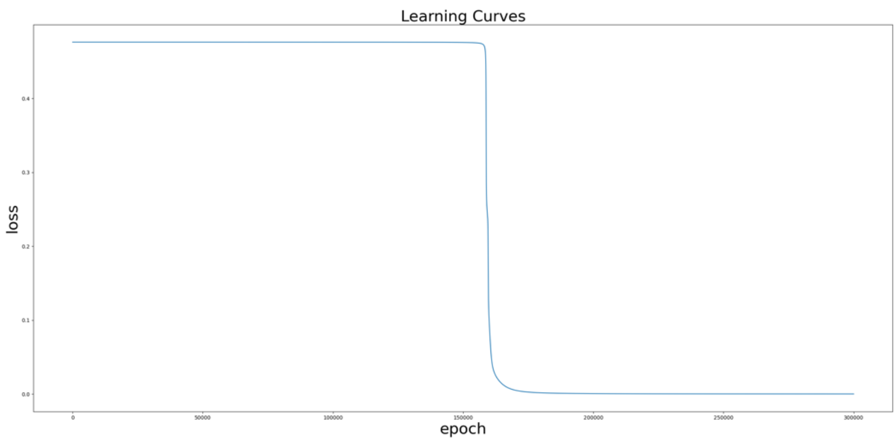




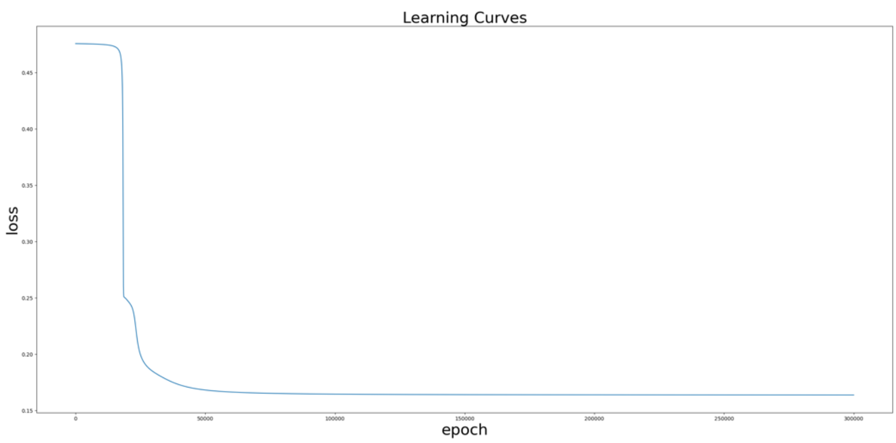
B. Try different numbers of hidden units

在 linear 資料中 hidden unit size 對訓練沒有顯著影響，而在 XOR 中較大的 hidden unit 會讓訓練時間變得更長，而且 Loss function 變得複雜會產生很多 local minimum，模型比較不好收斂，若是訓練資料沒那麼複雜可以盡量減少 hidden unit size。

XOR data set, hidden unit size = 10, LR = 0.1 :



XOR data set, hidden unit size = 2, LR = 0.1 :



C. Try without activation functions

No activation functions 對 linear data set 沒顯著影響，但會導致 XOR data set 之 Loss 沒辦法收斂。

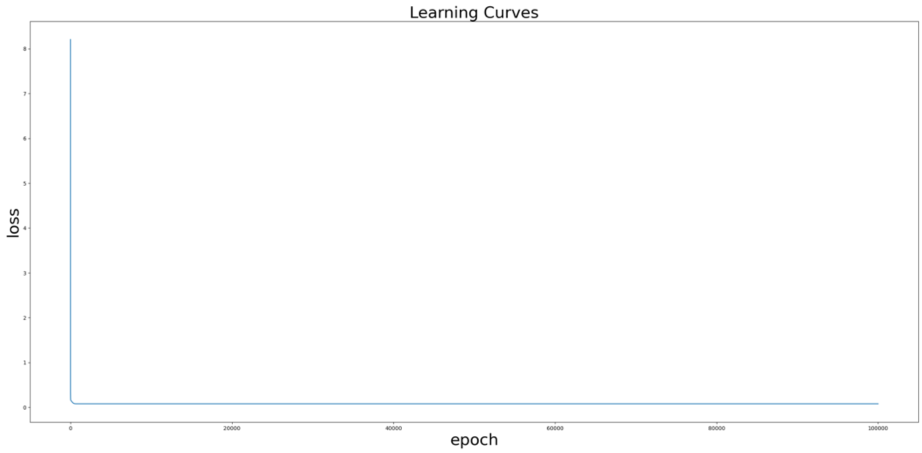
Linear data set :

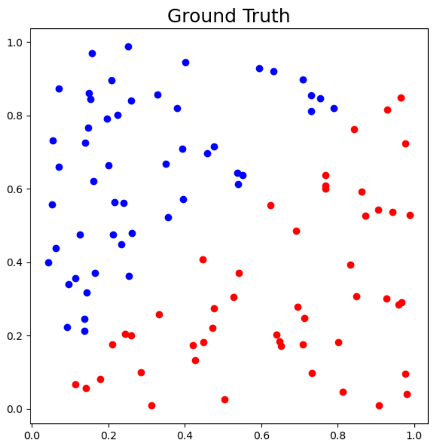
```
x2, y2 = generate_linear()
model = MLP(hidden_size = 3,activate = "none", learning_rate = 1e-2)
model.train(x2, y2, epoch = 100000)
```

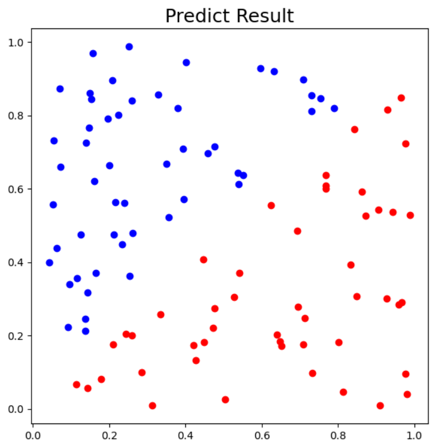
✓ 3.0s

```
epoch  0  loss :  8.20662186570062
epoch  5000  loss :  0.07671048204590875
epoch  10000  loss :  0.07671048204590869
epoch  15000  loss :  0.07671048204590869
epoch  20000  loss :  0.07671048204590869
epoch  25000  loss :  0.07671048204590869
epoch  30000  loss :  0.07671048204590869
epoch  35000  loss :  0.07671048204590869
epoch  40000  loss :  0.07671048204590869
epoch  45000  loss :  0.07671048204590869
epoch  50000  loss :  0.07671048204590869
epoch  55000  loss :  0.07671048204590869
epoch  60000  loss :  0.07671048204590869
epoch  65000  loss :  0.07671048204590869
epoch  70000  loss :  0.07671048204590869
epoch  75000  loss :  0.07671048204590869
epoch  80000  loss :  0.07671048204590869
epoch  85000  loss :  0.07671048204590869
epoch  90000  loss :  0.07671048204590869
epoch  95000  loss :  0.07671048204590869
```

```
Iter 1  |   Ground truth:  [1]  |   prediction:  [0.8335933]  |
Iter 2  |   Ground truth:  [0]  |   prediction:  [0.43895091]  |
Iter 3  |   Ground truth:  [1]  |   prediction:  [1.29271098]  |
Iter 4  |   Ground truth:  [0]  |   prediction:  [-0.37490809]  |
Iter 5  |   Ground truth:  [0]  |   prediction:  [-0.14062501]  |
Iter 6  |   Ground truth:  [0]  |   prediction:  [0.44198193]  |
Iter 7  |   Ground truth:  [0]  |   prediction:  [0.00292047]  |
Iter 8  |   Ground truth:  [1]  |   prediction:  [0.59003549]  |
Iter 9  |   Ground truth:  [0]  |   prediction:  [0.41623762]  |
Iter 10  |   Ground truth:  [1]  |   prediction:  [0.57916181]  |
Iter 11  |   Ground truth:  [1]  |   prediction:  [1.18522516]  |
Iter 12  |   Ground truth:  [1]  |   prediction:  [0.81457492]  |
Iter 13  |   Ground truth:  [0]  |   prediction:  [0.45146687]  |
Iter 14  |   Ground truth:  [0]  |   prediction:  [0.01406772]  |
Iter 15  |   Ground truth:  [1]  |   prediction:  [0.60958108]  |
Iter 16  |   Ground truth:  [1]  |   prediction:  [1.07461808]  |
Iter 17  |   Ground truth:  [1]  |   prediction:  [0.54477349]  |
Iter 18  |   Ground truth:  [1]  |   prediction:  [0.80770737]  |
Iter 19  |   Ground truth:  [1]  |   prediction:  [1.16133821]  |
Iter 20  |   Ground truth:  [1]  |   prediction:  [0.67034607]  |
Iter 21  |   Ground truth:  [1]  |   prediction:  [1.10986141]  |
Iter 22  |   Ground truth:  [0]  |   prediction:  [0.16858904]  |
Iter 23  |   Ground truth:  [1]  |   prediction:  [0.9434397]  |
Iter 24  |   Ground truth:  [1]  |   prediction:  [1.18411967]  |
Iter 25  |   Ground truth:  [0]  |   prediction:  [0.02986747]  |
...
Iter 98  |   Ground truth:  [0]  |   prediction:  [0.15470842]  |
Iter 99  |   Ground truth:  [1]  |   prediction:  [0.6451482]  |
Iter 100  |   Ground truth:  [0]  |   prediction:  [-0.27126481]  |
loss= 0.07671048204590869  accuracy= [100.] %
```







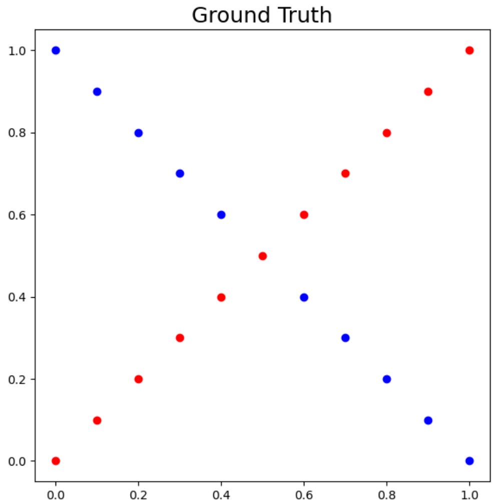
XOR data set :

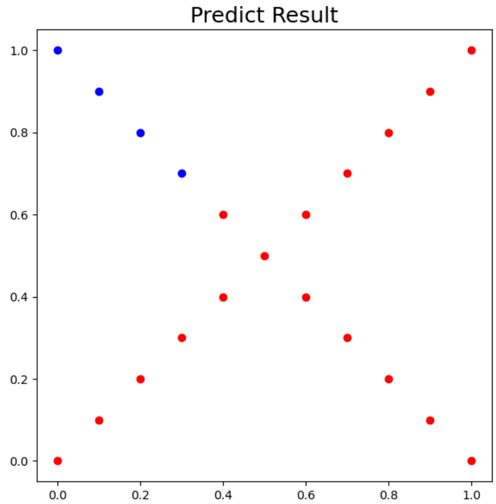
```
x2, y2 = generate_XOR_easy()
model = MLP(hidden_size = 2,activate = "none", learning_rate = 1e-4)
model.train(x2, y2, epoch = 100000)
```

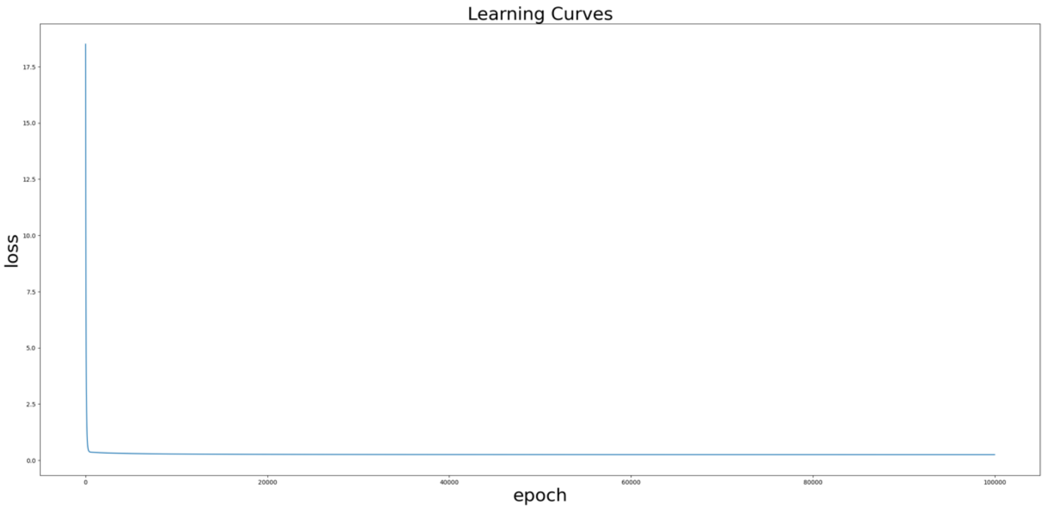
✓ 2.7s

```
epoch  0  loss :  18.499734411140444
epoch  5000  loss :  0.3005989787344996
epoch  10000  loss :  0.2785285295670828
epoch  15000  loss :  0.2685691535110954
epoch  20000  loss :  0.263078176805201 6
epoch  25000  loss :  0.2596718110630188
epoch  30000  loss :  0.2573889479728740 4
epoch  35000  loss :  0.2557740854886236
epoch  40000  loss :  0.2545856612127589 4
epoch  45000  loss :  0.2536844629623103 4
epoch  50000  loss :  0.2529849491809501 6
epoch  55000  loss :  0.2524318089540722 5
epoch  60000  loss :  0.2519877670745299 7
epoch  65000  loss :  0.2516268301227043
epoch  70000  loss :  0.2513303473705192
epoch  75000  loss :  0.2510846109589962
epoch  80000  loss :  0.2508793382075476
epoch  85000  loss :  0.2507066801979811 5
epoch  90000  loss :  0.2505605554985697 6
epoch  95000  loss :  0.2504361910324023 3
```

```
Iter 1  |    Ground truth:  [0]  |   prediction:  [0.4998498]  |
Iter 2  |    Ground truth:  [1]  |   prediction:  [0.54280804]  |
Iter 3  |    Ground truth:  [0]  |   prediction:  [0.4960906]  |
Iter 4  |    Ground truth:  [1]  |  Save As .ction:  [0.53045719]  |
Iter 5  |    Ground truth:  [0]  |   prediction:  [0.4923314]  |
Iter 6  |    Ground truth:  [1]  |   prediction:  [0.51810634]  |
Iter 7  |    Ground truth:  [0]  |   prediction:  [0.4885722]  |
Iter 8  |    Ground truth:  [1]  |   prediction:  [0.5057555]  |
Iter 9  |    Ground truth:  [0]  |   prediction:  [0.484813]  |
Iter 10  |    Ground truth:  [1]  |   prediction:  [0.49340465]  |
Iter 11  |    Ground truth:  [0]  |   prediction:  [0.4810538]  |
Iter 12  |    Ground truth:  [0]  |   prediction:  [0.4772946]  |
Iter 13  |    Ground truth:  [1]  |   prediction:  [0.46870295]  |
Iter 14  |    Ground truth:  [0]  |   prediction:  [0.4735354]  |
Iter 15  |    Ground truth:  [1]  |   prediction:  [0.4563521]  |
Iter 16  |    Ground truth:  [0]  |   prediction:  [0.4697762]  |
Iter 17  |    Ground truth:  [1]  |   prediction:  [0.44400125]  |
Iter 18  |    Ground truth:  [0]  |   prediction:  [0.466017]  |
Iter 19  |    Ground truth:  [1]  |   prediction:  [0.4316504]  |
Iter 20  |    Ground truth:  [0]  |   prediction:  [0.4622578]  |
Iter 21  |    Ground truth:  [1]  |   prediction:  [0.41929956]  |
loss= 0.2503298182594262  accuracy= [71.42857143] %
```







5. Extra (10%)

B. Implement different activation functions. (3%)

實作了 ReLU，放在模型中可以得到更低的 Loss：

```
def ReLU(x):
    return np.where(x > 0, x, 0)

def derivative_ReLU(x):
    return np.where(x > 0, 1, 0)
```

實驗結果：

```
x2, y2 = generate_linear()
model = MLP(hidden_size = 3,activate = "ReLU", learning_rate = 0.1)
model.train(x2, y2, epoch = 100000)
```

✓ 4.5s

```
epoch  0  loss :  0.24411503977137383
epoch  5000  loss :  0.0015091766636624881
epoch  10000  loss :  0.00014231161130480563
epoch  15000  loss :  7.992985870763848e-06
epoch  20000  loss :  3.362061168853767e-07
epoch  25000  loss :  1.3194666365675058e-08
epoch  30000  loss :  5.134289847974473e-10
epoch  35000  loss :  1.97769465823814e-11
epoch  40000  loss :  7.624948441579436e-13
epoch  45000  loss :  2.9849661007251094e-14
epoch  50000  loss :  1.14764727083931e-15
epoch  55000  loss :  4.4204705359551676e-17
epoch  60000  loss :  1.7168152847295326e-18
epoch  65000  loss :  6.668719662989465e-20
epoch  70000  loss :  2.563482806332538e-21
epoch  75000  loss :  9.877692680566904e-23
epoch  80000  loss :  3.857267875159889e-24
epoch  85000  loss :  1.4849860415291742e-25
epoch  90000  loss :  5.735380054599444e-27
epoch  95000  loss :  8.26150113691776e-28
```

```
...
Iter 98  |    Ground truth:  [1]  |   prediction:  [1.]  |
Iter 99  |    Ground truth:  [0]  |   prediction:  [0.]  |
Iter 100  |    Ground truth:  [1]  |   prediction:  [1.]  |
loss= 5.060446564569967e-28  accuracy= [100.] %
```

```python
x2, y2 = generate_XOR_easy()
model = MLP(hidden_size = 5,activate = "ReLU", learning_rate = 0.01)
model.train(x2, y2, epoch = 100000)
```

```
epoch  0  loss :  2.777253850574109
epoch  5000  loss :  0.04435611895868108
epoch  10000  loss :  0.029009299040473478
epoch  15000  loss :  0.01922600850829182
epoch  20000  loss :  0.01079963190727733
epoch  25000  loss :  0.003328773687635289
epoch  30000  loss :  0.0005179445354492696
epoch  35000  loss :  5.4792939893598795e-05
epoch  40000  loss :  1.8500946222901952e-06
epoch  45000  loss :  2.5713404305531004e-08
epoch  50000  loss :  3.6251884289541887e-10
epoch  55000  loss :  4.995154407491181e-12
epoch  60000  loss :  6.811561241274081e-14
epoch  65000  loss :  9.647461334321907e-16
epoch  70000  loss :  1.3330858350417206e-17
epoch  75000  loss :  1.8527680946696747e-19
epoch  80000  loss :  2.589896886945452e-21
epoch  85000  loss :  3.599776512568734e-23
epoch  90000  loss :  4.986596781586126e-25
epoch  95000  loss :  8.843069117769322e-27
```

```
Iter 1  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 2  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 3  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 4  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 5  |   Ground truth:  [0]  |   prediction:  [6.99440506e-15]  |
Iter 6  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 7  |   Ground truth:  [0]  |   prediction:  [2.0539126e-14]  |
Iter 8  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 9  |   Ground truth:  [0]  |   prediction:  [3.51940699e-14]  |
Iter 10  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 11  |   Ground truth:  [0]  |   prediction:  [4.92939023e-14]  |
Iter 12  |   Ground truth:  [0]  |   prediction:  [6.29496455e-14]  |
Iter 13  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 14  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 15  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 16  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 17  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 18  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 19  |   Ground truth:  [1]  |   prediction:  [1.]  |
Iter 20  |   Ground truth:  [0]  |   prediction:  [0.]  |
Iter 21  |   Ground truth:  [1]  |   prediction:  [1.]  |
loss= 1.2763376189923517e-27  accuracy= [100.] %
```