# Report

312552052鄭博元

# Introduction

- 這次要實作的backpropagation首先要進行forward得出每一層的線性組合(由輸入x權重+偏差),根據隱藏層數決定最後在輸出層得出一個prediction值,利用此預測值-真實值得出的error再回推每一層(使用微分的方式)得出的梯度去更新權重,而後達到訓練的模型越來越接近輸入獲得的真實值

# Experiment setups

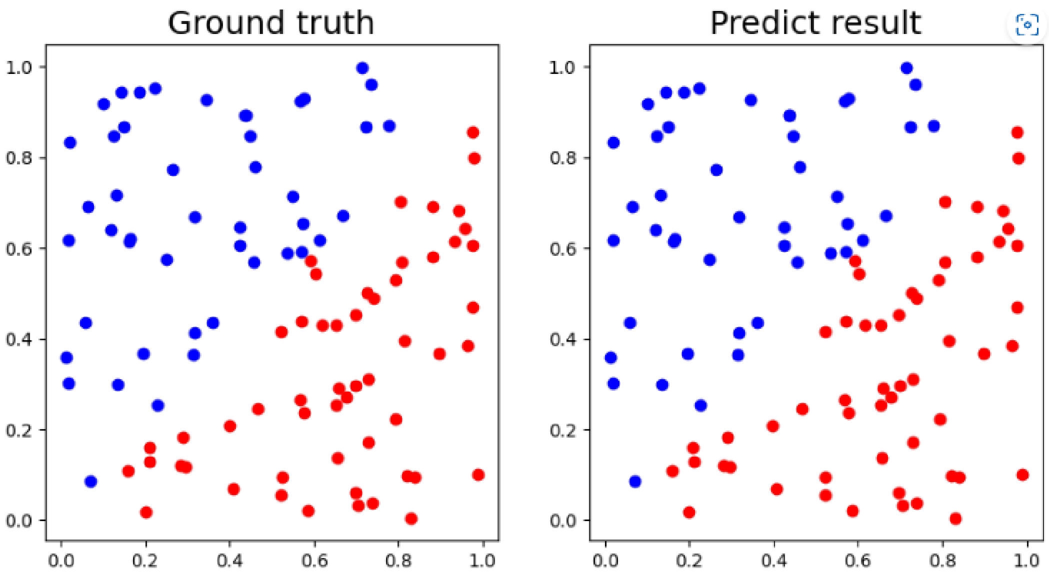
- Sigmoid functions:1 / (1 + np.exp(-x))

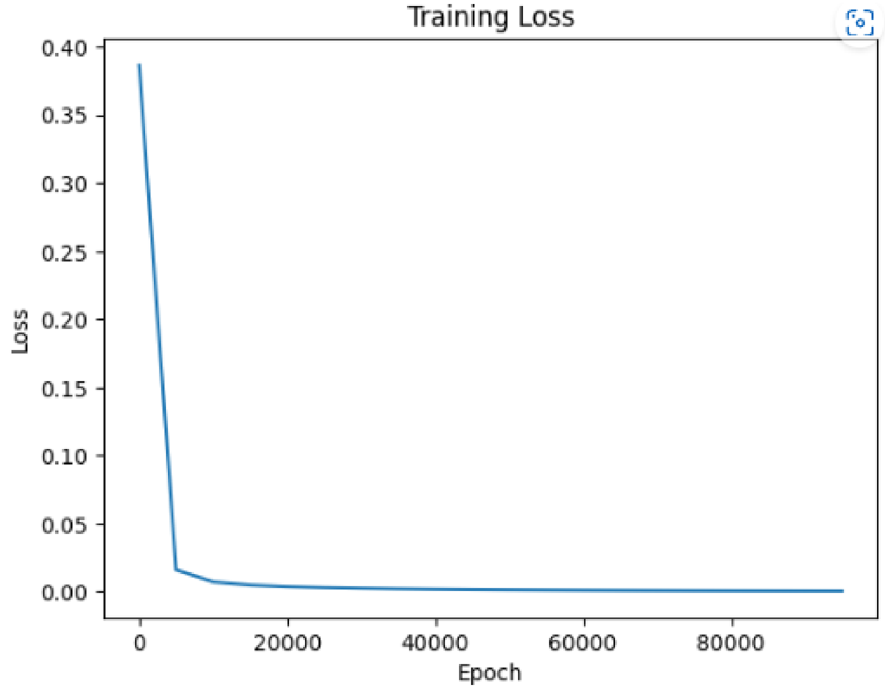
- Neural network:由initail,forward,backward,train,predict組成

- Backpropagation:由train中的forward以及backward去更新權重以最小化損失函數

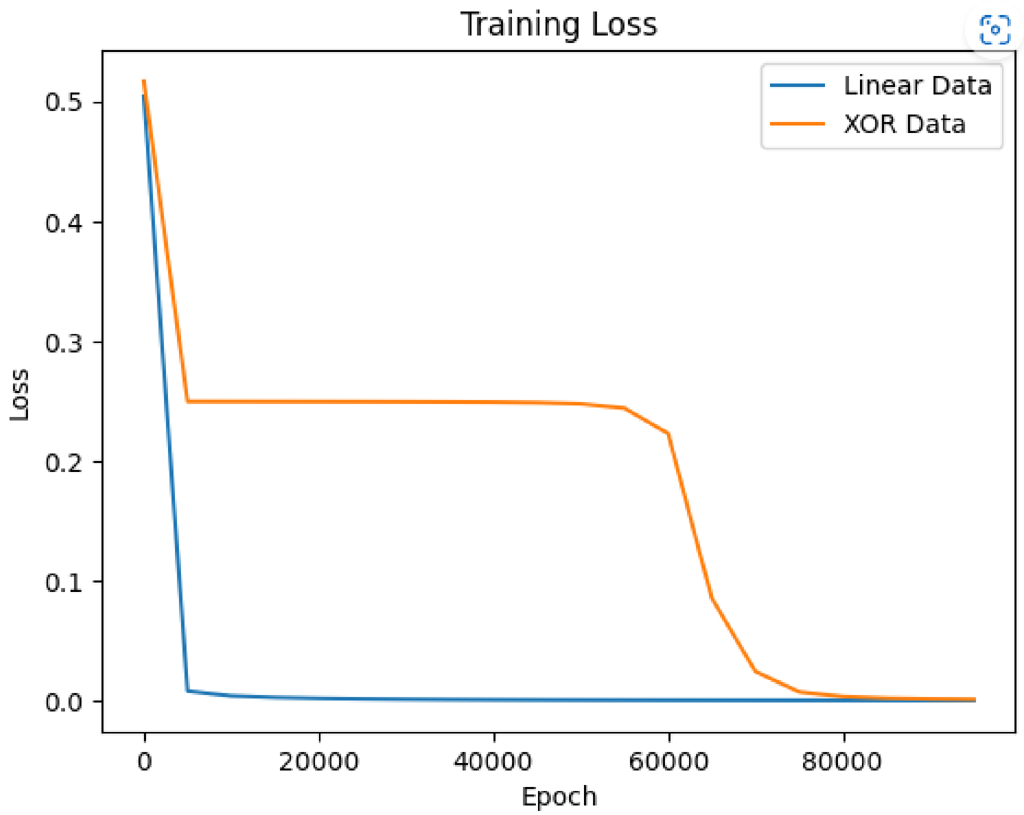
# Result of testing

```
Epoch 0, Loss: 0.24997016024732358, Prediction: [0.50064582]
Epoch 5000, Loss: 0.013694231419223718, Prediction: [0.9761997]
Epoch 10000, Loss: 0.006852356107366444, Prediction: [0.99733129]
Epoch 15000, Loss: 0.00471356451198783, Prediction: [0.99933328]
Epoch 20000, Loss: 0.0036104672451920555, Prediction: [0.99976434]
Epoch 25000, Loss: 0.0029193261854492886, Prediction: [0.99989796]
Epoch 30000, Loss: 0.0024335767601327634, Prediction: [0.99994952]
Epoch 35000, Loss: 0.0020661710018006732, Prediction: [0.99997256]
Epoch 40000, Loss: 0.0017749334570938311, Prediction: [0.99998398]
Epoch 45000, Loss: 0.001537263115284767, Prediction: [0.9999901]
Epoch 50000, Loss: 0.0013399084351415215, Prediction: [0.99999359]
Epoch 55000, Loss: 0.001174342321470686, Prediction: [0.99999567]
Epoch 60000, Loss: 0.0010345572801215097, Prediction: [0.99999697]
Epoch 65000, Loss: 0.0009160017639818555, Prediction: [0.99999782]
Epoch 70000, Loss: 0.0008150617499362906, Prediction: [0.99999838]
Epoch 75000, Loss: 0.000728795407262508, Prediction: [0.99999877]
Epoch 80000, Loss: 0.0006547817332585844, Prediction: [0.99999904]
Epoch 85000, Loss: 0.0005910207296522972, Prediction: [0.99999924]
Epoch 90000, Loss: 0.0005358587623937184, Prediction: [0.99999939]
Epoch 95000, Loss: 0.0004879284584603338, Prediction: [0.9999995]

Testing Results:
Iter91 | Ground truth: 1 | prediction: 1.00000
Iter92 | Ground truth: 0 | prediction: 0.00101
Iter93 | Ground truth: 1 | prediction: 1.00000
Iter94 | Ground truth: 1 | prediction: 1.00000
Iter95 | Ground truth: 0 | prediction: 0.00000
Iter96 | Ground truth: 0 | prediction: 0.00000
Iter97 | Ground truth: 1 | prediction: 1.00000
Iter98 | Ground truth: 1 | prediction: 0.95345
Iter99 | Ground truth: 1 | prediction: 1.00000
Iter100 | Ground truth: 1 | prediction: 0.99999

Final loss: 0.00045, accuracy: 100.00%
```



# Result of testing




```
Testing Results (XOR Data):
Iter91 | Ground truth: 0 | prediction: 0.00254
Iter92 | Ground truth: 1 | prediction: 0.99739
Iter93 | Ground truth: 0 | prediction: 0.00716
Iter94 | Ground truth: 1 | prediction: 0.99721
Iter95 | Ground truth: 0 | prediction: 0.01836
Iter96 | Ground truth: 1 | prediction: 0.99667
Iter97 | Ground truth: 0 | prediction: 0.03489
Iter98 | Ground truth: 1 | prediction: 0.99430
Iter99 | Ground truth: 0 | prediction: 0.04620
Iter100 | Ground truth: 1 | prediction: 0.93148

Final loss (Linear Data): 0.00014, accuracy: 100.00%

Final loss (XOR Data): 0.00085, accuracy: 100.00%
```

# Disscussion

- Learning rate若太低會導致需要很大量的迭代才能接近最優解

而太高會導致收斂不穩定錯過最優解

- 越多hidden units能使學習模式更複雜,但會耗費更多資源

而少的好處則是減少資源需求與訓練時間

- 沒有激活函數會導致皆為上一次輸出的線性組合,多少層都是同樣的線性並無變化,無法處理複雜的圖形與函數