back-propagation

Introduction

在這個實驗中，我設計了一神經網絡，可以用來對多種二維數據點的圖形做二元分類，程式中給定的訓練和測試資料集中包含三種二維數據點圖形:

1.Linear 2.XOR 3.Circle

該神經網絡為一個全連接的神經網絡，具有兩個隱藏層，用於二維數據點的二元分類。 兩個都將實現前向和後向傳導。

經實驗，該神經網路對二維數據點的圖形做二元分類，再經約15800次的訓練後，其準確率可達100%(以Linear圖形為例，learning rates=0.1)

本報告將詳細介紹該神經網路製作過程及實驗結果

Experiment setups

A.Sigmoid functions

在該神經網路鐘用的神經元碁活函數為sigmoid函數。

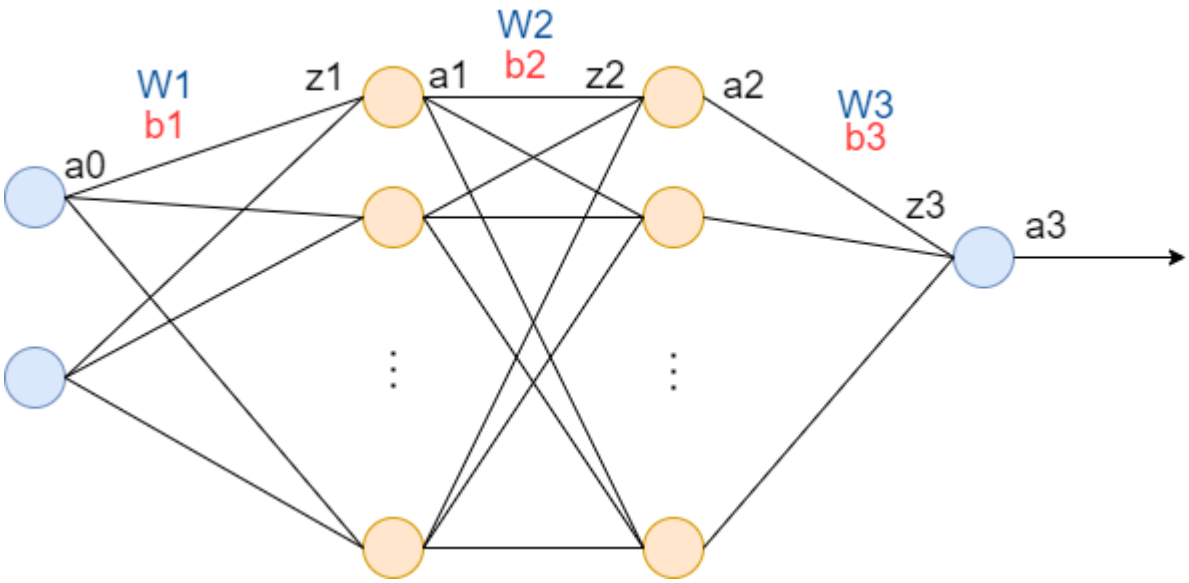
sigmoid 函數的導數:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

sigmoid 函數的導數:

```
def der_sigmoid(y):
    return y * (1.0 - y)
```

B. Neural network



神經網絡定義如上圖

註釋：

hidden_size:為1x2的Array，用於控制隱藏層的神經元數量。

X:圖形的二維輸入數據集，為nx2的Array，其中n是該圖形的數據數量。

Y:該圖形的分類標籤，為nx1的Array，其中n是該圖形的數據數量。

Wi:為第i層和第i-1層的權重矩陣，其中W1、W2 和 W3 的大小分別為:

W1=hidden_size[0]x2

W2=hidden_size[1]xhidden_size[0]

W3=1xhidden_size[1]

bi:與第i層相關的偏置質向量。

ai:第i層神經元的激活向量。

zi:第i層神經元的加權輸出向量。

模型參數:

```
self.W1 = np.random.randn(hidden_size[0], 2)
self.W2 = np.random.randn(hidden_size[1], hidden_size[0])
self.W3 = np.random.randn(1, hidden_size[1])

self.b1 = np.ones((hidden_size[0],1))
self.b2 = np.ones((hidden_size[1],1))
self.b3 = np.ones((1,1))
```

向前傳導:

```
 def forward(self, inputs):
    self.a0 = inputs.T
    self.z1 = np.add( np.dot( self.W1, self.a0 ), self.b1 )
    self.a1 = sigmoid(self.z1)
    self.z2 = np.add( np.dot( self.W2, self.a1 ), self.b2 )
    self.a2 = sigmoid(self.z2)
    self.z3 = np.add( np.dot( self.W3, self.a2 ), self.b3 )
    self.a3 = sigmoid(self.z3)
    return self.a3.T
```

C. Backpropagation

向後傳導:

```
def backward(self):
    self.grad_z3 = self.error.T
    self.grad_z2 = np.dot (self.W3.T, self.grad_z3) * der_sigmoid(self.a2)
    self.grad_z1 = np.dot (self.W2.T, self.grad_z2) * der_sigmoid(self.a1)
    self.grad_W3 = np.dot(self.grad_z3, self.a2.T)
    self.grad_W2 = np.dot(self.grad_z2, self.a1.T)
    self.grad_W1 = np.dot(self.grad_z1, self.a0.T)

    self.grad_b3 = np.sum(self.grad_z3)
    self.grad_b2 = np.sum(self.grad_z2)
    self.grad_b1 = np.sum(self.grad_z1)
```

更新參數:

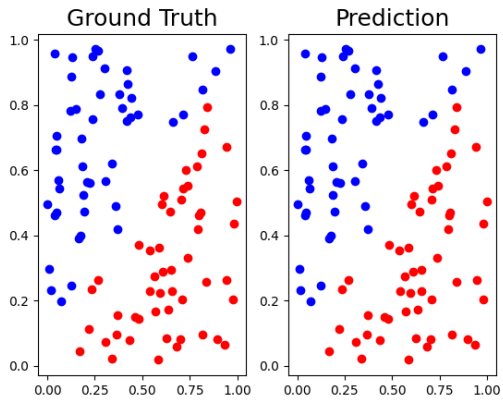
```
 def update(self, alpha):
    self.W1 -= alpha * self.grad_W1
    self.W2 -= alpha * self.grad_W2
    self.W3 -= alpha * self.grad_W3
    self.b1 -= alpha * self.grad_b1
    self.b2 -= alpha * self.grad_b2
    self.b3 -= alpha * self.grad_b3
```

```
    self.b3 -= alpha * self.grad_b3
    self.grad_W1 = 0
    self.grad_W1 = 0
    self.grad_W2 = 0
    self.grad_W3 = 0
    self.grad_b1 = 0
    self.grad_b2 = 0
    self.grad_b3 = 0
```

Results of my testing

1.Linear:

A. Screenshot and comparison figure




```
[[9.99997435e-01]
 [1.18909471e-05]
 [9.98593544e-01]
 [7.46260682e-06]
 [6.91751536e-06]
 [1.01030106e-05]
 [9.99997784e-01]
 [9.70046456e-06]
 [7.94252530e-04]
 [9.99993153e-01]
 [9.78125160e-01]
 [9.99996093e-01]
 [7.03482442e-06]
 [9.99997607e-01]
 [9.99995673e-01]
 [3.20973265e-05]
 [9.99997754e-01]
 [9.99996791e-01]
 [1.40992625e-05]
 [6.82956352e-06]
 [6.81340158e-06]
 [9.85887364e-01]
 [7.76225824e-06]
 [6.50061787e-02]
 [5.27139498e-05]
 [9.99997584e-01]
 [1.06134141e-05]
```

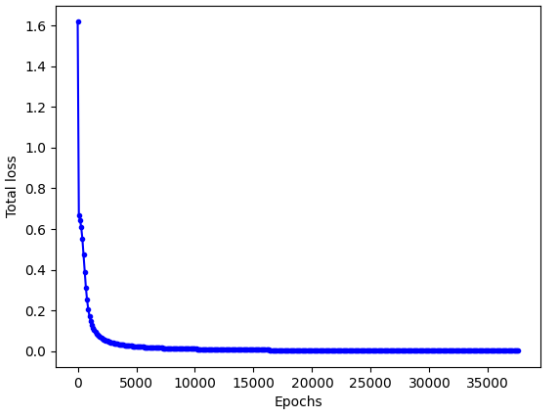
```
[6.60040562e-06]
[9.99996611e-01]
[9.93114688e-01]
[2.09949603e-02]
[8.31597342e-06]
[9.99997452e-01]
[9.99979857e-01]
[9.99997772e-01]
[9.99996801e-01]
[7.34719065e-06]
[6.81708454e-06]
[9.99995362e-01]
[6.66433680e-06]
[6.56205643e-06]
[9.99994964e-01]
[9.99978826e-01]
[2.76049019e-05]
[9.99997466e-01]
[4.39752437e-05]
[3.47701139e-05]
[9.99997905e-01]
[9.99676951e-01]
[9.99997265e-01]
[9.98767846e-01]
[9.99997566e-01]
[9.99997759e-01]
[1.66395358e-05]
[9.99827182e-01]
[9.63699455e-06]
[7.04011601e-06]
[9.99997083e-01]
[7.90260452e-06]
[7.27579350e-06]
[8.79892346e-06]
[9.05533329e-06]
[9.99994808e-01]
[9.99996097e-01]
[4.65408164e-05]
[9.99957532e-01]
[9.99997299e-01]
[8.25239418e-06]
[2.37538057e-05]
[9.99994723e-01]
[9.99991390e-01]
[1.02492727e-05]
[8.43929177e-06]
[9.99996513e-01]
[9.99997614e-01]
[9.35983234e-06]
[9.99988574e-01]
[9.99997195e-01]
[1.02970964e-05]
[1.11341026e-04]
[9.99997001e-01]
```

```
[9.98049802e-01]
[9.99995554e-01]
[1.63712783e-05]
[6.68809907e-06]
[8.87336188e-06]
[8.75465268e-06]
[2.25158491e-02]
[7.67563971e-05]
[9.99989394e-01]
[9.39206658e-01]
[4.49750850e-04]
[7.82398841e-06]
[9.99997845e-01]
[3.70404887e-05]
[1.44935838e-04]
[9.99996512e-01]
[9.99997035e-01]
[2.31353901e-04]
[9.99991190e-01]]
```

B. Show the accuracy of your prediction

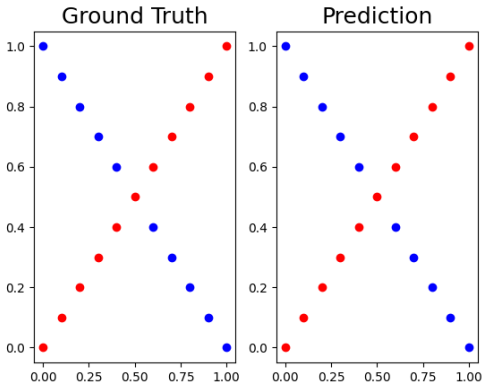
```
accuracy: 99.78%, loss: 0.002248966286134061
```

C. Learning curve (loss, epoch curve)



2.XOR

A. Screenshot and comparison figure

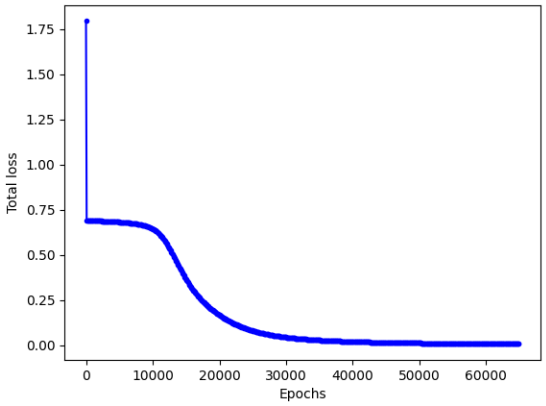


```
[[0.00962991]
 [0.99512548]
 [0.0090736 ]
 [0.99502686]
 [0.00856444]
 [0.99486749]
 [0.00809992]
 [0.99445085]
 [0.00767672]
 [0.97393059]
 [0.00729119]
 [0.0069399 ]
 [0.96742977]
 [0.00661994]
 [0.99783698]
 [0.00632909]
 [0.99819354]
 [0.00606585]
 [0.99820368]
 [0.00582928]
 [0.99817891]]
```

B. Show the accuracy of your prediction

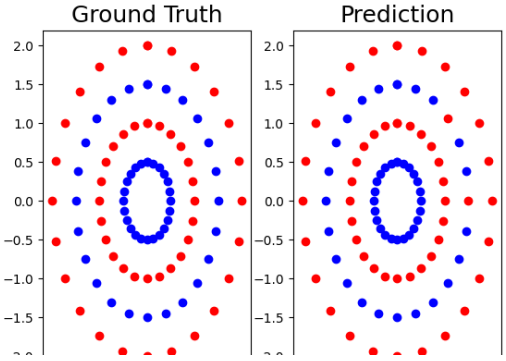
```
accuracy: 99.20%, loss: 0.008101910184258035
```

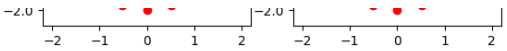
C. Learning curve (loss, epoch curve)



3.Circle

A. Screenshot and comparison figure





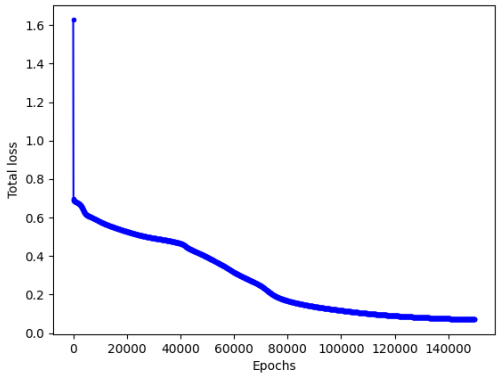

```
[[5.83982198e-09]
 [7.15097074e-03]
 [9.86091332e-01]
 [9.99992487e-01]
 [9.87860181e-09]
 [2.10164212e-02]
 [9.71601340e-01]
 [9.99996415e-01]
 [1.08552716e-01]
 [4.15446610e-02]
 [9.65943556e-01]
 [9.99996701e-01]
 [6.36832611e-02]
 [1.00486819e-01]
 [3.55288917e-01]
 [9.99992126e-01]
 [7.26848629e-02]
 [1.85387910e-01]
 [8.82995095e-01]
 [9.99723746e-01]
 [1.80127878e-01]
 [4.88533679e-01]
 [8.35241076e-01]
 [9.84879354e-01]
 [2.63301346e-01]
 [1.44488334e-01]
 [1.51744665e-01]
 [9.65066495e-01]
 [3.62928277e-01]
 [2.49968084e-02]
 [7.92498808e-01]
 [9.94150638e-01]
 [4.56446950e-02]
 [1.36245100e-01]
 [9.52936536e-01]
 [9.99942246e-01]
 [2.00576146e-02]
 [1.66901393e-01]
 [8.61820027e-01]
 [9.99995445e-01]
 [2.15052395e-02]
 [1.10553277e-03]
 [8.24730267e-01]
 [9.99998149e-01]
 [6.57245052e-04]
 [1.10227599e-02]
 [9.87860298e-01]
 [9.99994771e-01]
 [9.27843695e-04]
 [1.66989855e-02]
```

```
 [9.99965105e-01]
 [9.99661101e-01]
 [2.93185456e-04]
 [4.96961881e-04]
 [9.94598022e-01]
 [9.99432984e-01]
 [1.11233396e-02]
 [3.97124623e-04]
 [9.86187875e-01]
 [9.99970608e-01]
 [9.99547421e-03]
 [5.82307555e-04]
 [9.98249154e-01]
 [9.99994852e-01]
 [1.96692345e-03]
 [7.43838358e-03]
 [9.94943096e-01]
 [9.99996518e-01]
 [8.28135780e-03]
 [4.04842808e-03]
 [9.95859062e-01]
 [9.99996673e-01]
 [4.92921391e-04]
 [1.18539955e-02]
 [9.96115808e-01]
 [9.99996187e-01]
 [4.15939442e-06]
 [1.78431682e-03]
 [9.82233584e-01]
 [9.99993264e-01]
 [7.04076190e-06]
 [4.05440041e-05]
 [9.98461992e-01]
 [9.99971724e-01]
 [3.99761183e-04]
 [2.96285389e-06]
 [9.99999816e-01]
 [9.99713463e-01]
 [3.58409183e-04]
 [5.81546912e-03]
 [9.99999990e-01]
 [9.99615750e-01]
 [2.06683089e-06]
 [1.67175413e-04]
 [9.96592597e-01]
 [9.99959546e-01]
 [5.83982198e-09]
 [7.15097074e-03]
 [9.86091332e-01]
 [9.99992487e-01]]
```

B. Show the accuracy of your prediction

```
accuracy: 94.88%, loss: 0.0703282816455223
```

C. Learning curve (loss, epoch curve).



Discussion

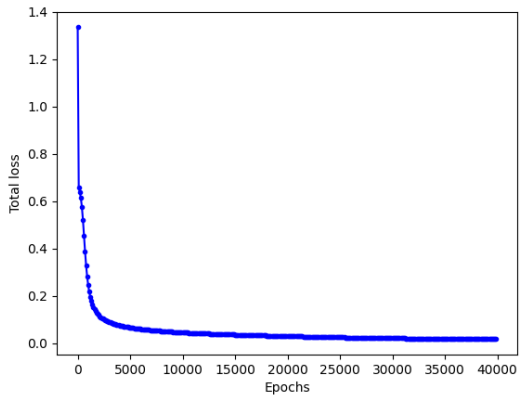
A.Try different learning rates

以下將測試三種不同的learning rates。

這邊的實驗以Linear圖形為主，分別以0.001、0.5、0.999三種不同的學習率來分析在經40000次的訓練後，學習曲線的變化。

1.learning rates=0.001

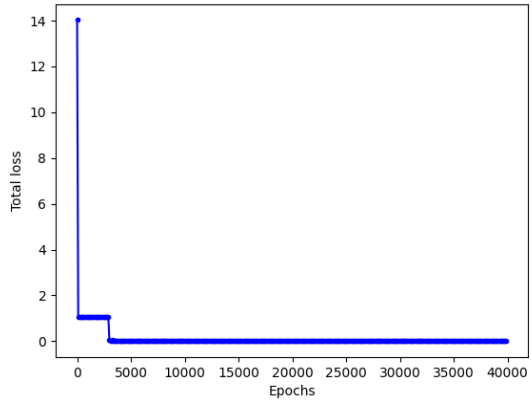
Learning curve:




```
accuracy: 98.50%, loss: 0.0177474342336609
```

2.learning rates=0.5

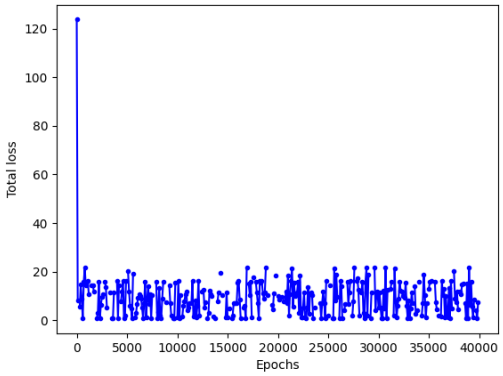
Learning curve:



```
accuracy: 100.00%, loss: 1.3563883000373575e-05
```

3.learning rates=0.999

Learning curve:




```
accuracy: 45.00%, loss: 13.32723716901954
```

learning rates=0.001 約在訓練35000次後收斂，準確率為98.50%

learning rates=0.1 約在訓練5000次後收斂，準確率為100%

learning rates=0.999 無法收斂，準確率為45.00%

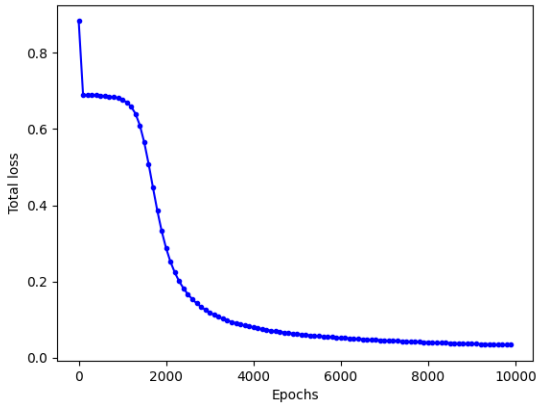
在以上的實驗結果中不難發現過大的learning rates容易導致無法收斂，過小的學習率也會造成收斂時間過長，甚至無法有最好的收斂。因此挑選合適的learning rates對神經網路而言相當重要。

B.different numbers of hidden units

以下分別測試在learning rates同為0.001，訓練次數為10000次的情況下，2x2、10x10、30x30的隱藏層對神經網路的影響

1.hidden layer = 2x2

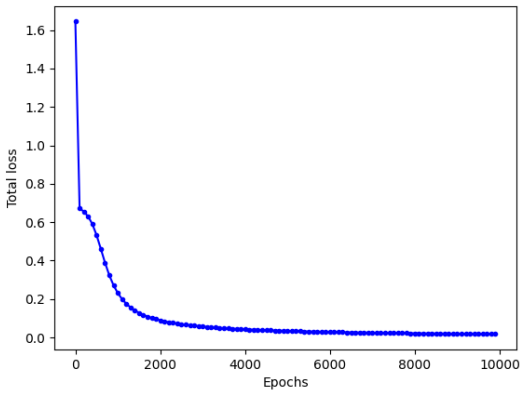
Learning curve:




```
accuracy: 97.08%, loss: 0.033850269071221566
```

2.hidden layer = 10x10

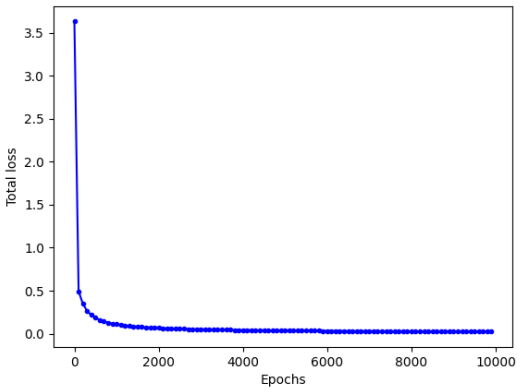
Learning curve:




```
accuracy: 98.43%, loss: 0.01748660496365463
```

2.hidden layer = 30x30

Learning curve:




```
accuracy: 97.80%, loss: 0.02669361139844492
```

經由結果可得出，隱藏層節點數量少，收斂的速度較慢，優點是訓練速度快，也不必占用過多的資源。而隱藏層節點數量多，收斂的速度較快，缺點是訓練速度慢，也占用較多的資源，另外還有可能造成過擬合的效應。因此視資料類型選擇合適的隱藏層和節點數量亦是相當重要！

Extra

Implement different activation functions

以下實現用tanh**函數作為**神經網路鐘用的神經元碁活函數

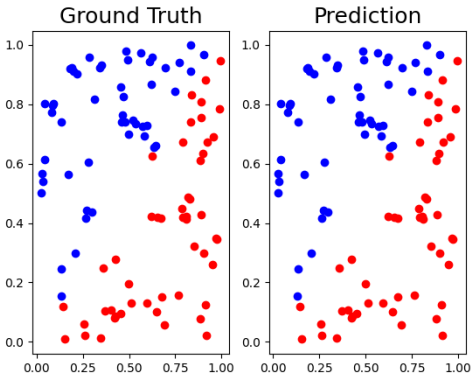
tanh函數的導數:

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

tanh函數的導數:

```
def der_tanh(y):
    return 1-tanh(y)*tanh(y)
```

以Linear的圖形分類為例:




```
accuracy: 98.89%
```