一、前言

此實驗為神經網路倒傳遞實驗，內容為建構一個 784-128-10 的類神經網路，並計算神經網路反向傳播的梯度，用於更新權重，藉此訓練模型使神經網路準確率達 0.9 以上。

二、數學介紹

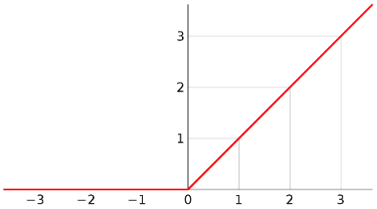
(一)One-hot Encode：

1. 我們可以用這個方法可以對標籤進行熱編碼轉換即 One-hot encode，這可以將標籤轉換為向量形式。
2. 利用「單位矩陣」乘上要轉換的標籤資料。

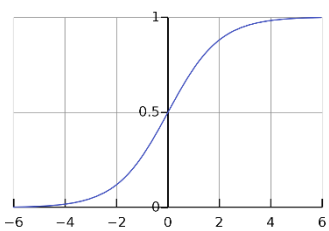
(二)Forward Pass：

這個使用在計算從輸入到輸出的過程，包含了加權、激活函數的應用以及預測結果。

1. ReLU：
   (1)公式為 f(x) = max(0,x)。
   (2)其中 x 是輸入信號，f(x)是 ReLU 函數的輸出。
   (3)ReLU 函數的特點是在輸入為正時，輸出等於輸入；當輸入為負時，輸出為 0。換句話說，ReLU 將負值的輸入全部截斷為 0，而對於正值則不做任何改變，這種非線性的轉換有助於神經網路學習更複雜的函數關係。



2. Sigmoid：
   (1)公式為 f(x) = 1/(1+e^(-x))。
   (2)其中 x 是輸入信號，f(x)是 Sigmoid 函數的輸出。
   (3)Sigmoid 函數的特點是它是一個 S 型曲線，當輸入趨近於正無窮大時，函數的值趨近於 1；當輸入趨近於負無窮大時，函數的值趨近於 0。



3. Sigmoid 的倒函數：

另 $y = \frac{1}{1+e^{-x}}$

$\frac{dy}{dx} = \frac{d}{dx}(\frac{1}{1+e^{-x}}) = \frac{d}{dy}(\frac{1}{y}) \cdot \frac{dy}{dx} = -\frac{1}{y^2} \cdot \frac{dy}{dx}$

即 $f'(x) = f(x)(1-f(x))$

因此 $\frac{dy}{dx} = y(1-y)$

(三)矩陣乘法

$$c_{nm} = A_{nm} \cdot B_{nm}$$

$$\begin{bmatrix} C_{11} & C_{21} & \dots & C_{m1} \\ C_{12} & C_{22} & \dots & C_{m2} \\ \dots & \dots & C_{ji} & \dots \\ C_{1n} & C_{2n} & \dots & C_{nm} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} + \cdots + A_{1m}B_{n1} & \dots & A_{11}B_{1m} + A_{12}B_{2m} + \cdots + A_{1m}B_{nm} \\ \dots & \dots & \dots \\ A_{n1}B_{11} + A_{n2}B_{21} + \cdots + A_{nm}B_{n1} & \dots & A_{n1}B_{1m} + A_{n2}B_{2m} + \cdots + A_{nm}B_{nm} \end{bmatrix}$$

三、程式碼介紹

首先，匯入 Numpy 模組。接著載入權重矩陣、標籤資料和輸入資料。其中，fc1 為 Input Layer 的權重矩陣，fc2 為隨機創建的(128,10)矩陣，是 Output Layer 的權重矩陣。載入輸入資料後，再將其進行正規化處理(除以 255)，並將輸入的圖像 reshape 轉成一維資料。

```
import numpy as np


# 載入權重矩陣
fc1=np.load('ANN0.npy')
fc2=np.random.randn(128*10).reshape([128,10])

# 載入標籤資料
label=np.load('mnistLabel.npy')

# 載入輸入資料
input=np.load('mnist.npy')/255.0
input=np.reshape(input,[len(input),-1])
```

接下來，將標籤進行 One-hot Encoding 後，建立一個空串列紀錄準確度。就可以開始進行模型迭代訓練。

```
# 將標籤進行 One-hot Encode
D = np.identity(10)[label]

#-----------------------------------------End code

# 建立準確度空字串
accuracy = []

# 開始進行迭代訓練
for ep in range(100):
```

X1 是我們用來計算第一層的輸出，由於輸入矩陣與權重矩陣的形狀無法進行矩陣乘法，因此先將 input 與 fc1 進行轉置，使用點乘後，再進行一次轉置，最後得到(60000,128)的矩陣。

```
# 向前迭代 X1 層
x1 = (np.dot(fc1.T,input.T)).T
```

將 X1 複製到 A1，並且使用激活函數 ReLU，將 A1 中的負值變為 0。

```
# 向前迭代 A1 層
A1 = x1.copy()
A1[A1<0] = 0
```

接著利用 X2 層計算第二層的輸出，方法、原理與第一層相同。

```
# 向前迭代 X2 層
x2 = (np.dot(A1,fc2))
```

將輸出 X2 進行 Sigmoid 激活函數，得到 A2。

```
# 向前迭代 A2 層
A2 = 1/(1+np.exp(-x2))
```

這行程式是用來計算反向傳播的誤差，利用 Sigmoid 的導數乘以(D-A2)，(D-A2)代表真實標籤與預測獲得的值的差。

```
#sigmoid derivative*predict value -->delta(loss)
    backward_delta = A2*(1-A2)*(D-A2)
```

A1 為前一層的激活值，將 A1 乘以負值使梯度呈現減少方向，再把結果轉置乘以 Backwrd_Delta。

```
grad = np.dot(((-2)*A1).T,backward_delta)
```

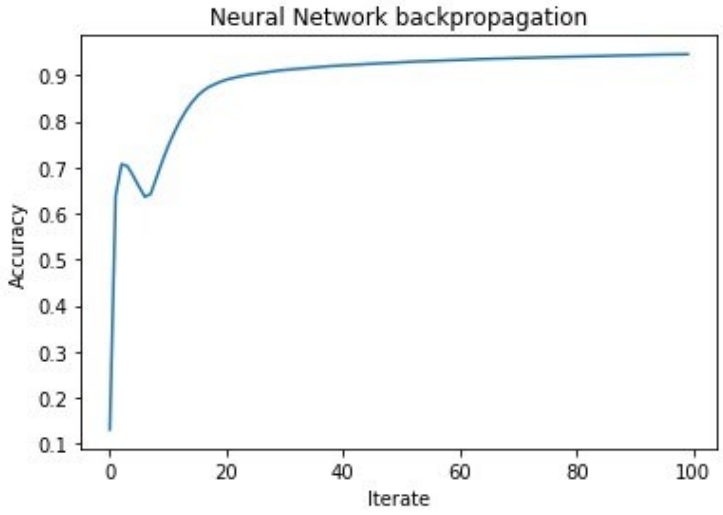
利用 choose=np.argmax(A2,1) 找到輸出層中最大的索引 print(str(ep+1)+": "+str(np.sum(choose==label)/len(label))) 將 choose=label 計算出正確的樣本數除以總樣本 label 得到準確率，再利用 accuracy.append(np.sum(choose == label) / len(label))來列出準確度。

```
fc2=fc2-0.05*grad

choose=np.argmax(A2,1)
print(str(ep+1)+": "+str(np.sum(choose==label)/len(label)))
accuracy.append(np.sum(choose == label) / len(label))
```

最後，進行學習準確度的時間曲線圖繪製，X 座標為迭代次數，Y 座標為準確度。

```
import matplotlib.pyplot as plt

plt.plot(accuracy)
plt.title("Neural Network backpropagation")
plt.xlabel("Iterate")
plt.ylabel("Accuracy")
plt.show()
```

四、學習準確度的時間曲線



五、結論(含分工表)

模型準確率的提高：從訓練過程中輸出的準確率數據可以觀察到，隨著訓練的進行，神經網路模型的準確率從初始值不斷提高，最終達到了 0.90875。這顯示了隨著訓練次數的增加，神經網路模型對手寫數字的識別能力逐漸增強。

分工表：

組員	負責項目
林郅恒	One-hot Encode Forward Pass 數學介紹 結論
王禹晴	前言 矩陣乘法介紹
何冠增	程式碼介紹 統整 Word
陳立堯	程式碼介紹
張耕碩	程式碼修改、介紹 學習準確度的時間曲線