Deep learning　lab1　　陳品翰　313552042　網工所碩一

1. Introduction

在這個作業中，我實作了一個 deep learning model，model 為兩個 hidden layer 的架構，每個 hidden layer 有 4 個 neuron，input layer size 為 2，output layer size 為 1，data set 為助教提供的兩個函式生成的，把兩個函式生成的 data set 放入模型後，用 gradient decent 的方法來做訓練，也就是實作 forward pass 和 backward pass 的部分，之後可以得到準確率超過 90%的訓練結果。

2. Implementation Details

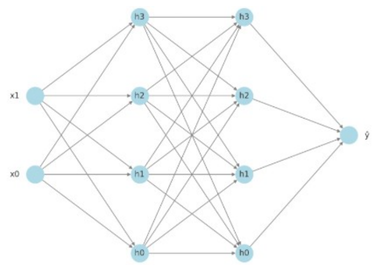
A. Sigmoid function

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

def sigmoid_derivative(x):
    #return 1
    return x * (1 - x)
```

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

附圖為 sigmoid function 的數學定義和其導數跟實作的程式碼

B. Neural network architecture



神經網路的架構為兩層 hidden layer，每個 hidden layer 有 4 個 neuron，當 data 進入 model 時，每過一層，會經過一個線性變換跟一個 activation function，所以這個神經網路中，data 總共會經過三次線性轉換跟三次 activation function

C. Back-propagation

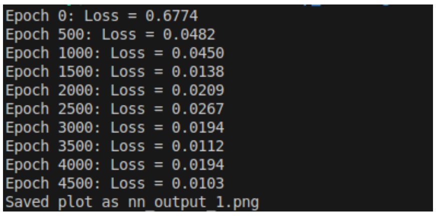
```
dA3 = cross_entropy_derivative(y_train , A3)
dZ3 = dA3 * sigmoid_derivative(A3)
dW3 = np.dot(A2.T, dZ3)
db3 = np.sum(dZ3, axis=0, keepdims=True)

dA2 = np.dot(dZ3, W3.T)
dZ2 = dA2 * sigmoid_derivative(A2)
dW2 = np.dot(A1.T, dZ2)
db2 = np.sum(dZ2, axis=0, keepdims=True)

dA1 = np.dot(dZ2, W2.T)
dZ1 = dA1 * sigmoid_derivative(A1)
dW1 = np.dot(X_train.T, dZ1)
db1 = np.sum(dZ1, axis=0, keepdims=True)
```

在 gradient decent 的過程中，由於要對前面的參數做偏微分時，會因為 chain rule 的關係而展開非常多項，這些展開的部分可以用 back propagation 來得到，之後再和 forward pass 得到的部分做相乘，來得到偏微分的結果。另外，在展開各層的偏微分算式時，可以發現相鄰的兩層 layer 之間要相乘的部分有很大的相似性，所以可以把 back propagation 的結果存下來，並且給上一層使用，如此一來便可以達到讓 gradient decent 更有效率的目的。

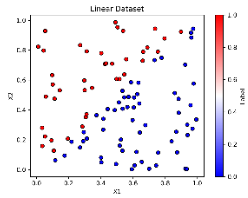
3. Experimental Result

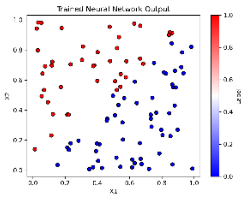
   A. Screenshot and comparison figure

   附圖為 generate_linear() 生成的 data set 的 training loss 和 ground truth/prediction 的圖片，中間的為 ground truth 右邊的為 prediction

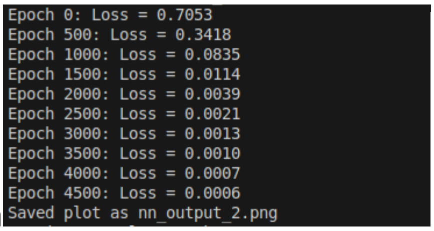


```
Epoch 0: Loss = 0.6774
Epoch 500: Loss = 0.0482
Epoch 1000: Loss = 0.0450
Epoch 1500: Loss = 0.0138
Epoch 2000: Loss = 0.0209
Epoch 2500: Loss = 0.0267
Epoch 3000: Loss = 0.0194
Epoch 3500: Loss = 0.0112
Epoch 4000: Loss = 0.0194
Epoch 4500: Loss = 0.0103
Saved plot as nn_output_1.png
```

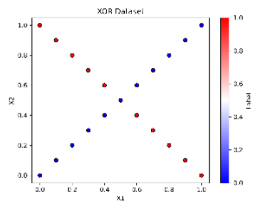


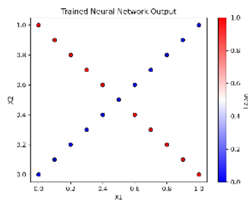


   附圖為 generate_XOR_easy() 生成的 data set 的 training loss 和 ground truth/prediction 的圖片，中間的為 ground truth 右邊的為 prediction



```
Epoch 0: Loss = 0.7053
Epoch 500: Loss = 0.3418
Epoch 1000: Loss = 0.0835
Epoch 1500: Loss = 0.0114
Epoch 2000: Loss = 0.0039
Epoch 2500: Loss = 0.0021
Epoch 3000: Loss = 0.0013
Epoch 3500: Loss = 0.0010
Epoch 4000: Loss = 0.0007
Epoch 4500: Loss = 0.0006
Saved plot as nn_output_2.png
```





   B. Show the accuracy of your prediction

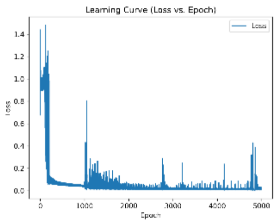
   左圖為 generate_linear()的結果，右圖為 generate_XOR_easy()的結果

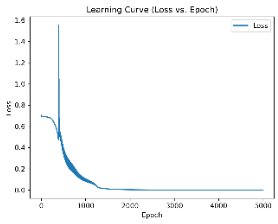
```
Iter90  Ground truth: 0  prediction: 0.0000
Iter91  Ground truth: 1  prediction: 0.9998
Iter92  Ground truth: 1  prediction: 0.9992
Iter93  Ground truth: 0  prediction: 0.0000
Iter94  Ground truth: 1  prediction: 0.9999
Iter95  Ground truth: 0  prediction: 0.0029
Iter96  Ground truth: 0  prediction: 0.0000
Iter97  Ground truth: 1  prediction: 0.9988
Iter98  Ground truth: 1  prediction: 0.9910
Iter99  Ground truth: 0  prediction: 0.0000
Loss: 0.0398  Accuracy: 97.0%
```

```
Iter11  Ground truth: 0  prediction: 0.0001
Iter12  Ground truth: 1  prediction: 0.9982
Iter13  Ground truth: 0  prediction: 0.0000
Iter14  Ground truth: 1  prediction: 1.0000
Iter15  Ground truth: 0  prediction: 0.0000
Iter16  Ground truth: 1  prediction: 1.0000
Iter17  Ground truth: 0  prediction: 0.0000
Iter18  Ground truth: 1  prediction: 1.0000
Iter19  Ground truth: 0  prediction: 0.0000
Iter20  Ground truth: 1  prediction: 0.9999
Loss: 0.0005  Accuracy: 100.0%
```

C. Learning curve

左圖為 generate_linear()的結果，右圖為 generate_XOR_easy()的結果





4. Discussion

A. Try different learning rates

左圖為 generate_linear()的結果，右圖為 generate_XOR_easy()的結果

```
Iter90  Ground truth: 1  prediction: 1.0000
Iter91  Ground truth: 0  prediction: 0.0043
Iter92  Ground truth: 1  prediction: 1.0000
Iter93  Ground truth: 0  prediction: 0.0008
Iter94  Ground truth: 0  prediction: 0.0000
Iter95  Ground truth: 0  prediction: 0.0000
Iter96  Ground truth: 0  prediction: 0.0129
Iter97  Ground truth: 1  prediction: 0.9158
Iter98  Ground truth: 0  prediction: 0.0000
Iter99  Ground truth: 0  prediction: 0.0000
Loss: 0.0123  Accuracy: 100.0%
```

```
Iter13  Ground truth: 0  prediction: 0.0934
Iter14  Ground truth: 1  prediction: 0.7866
Iter15  Ground truth: 0  prediction: 0.0188
Iter16  Ground truth: 1  prediction: 0.8783
Iter17  Ground truth: 0  prediction: 0.0060
Iter18  Ground truth: 1  prediction: 0.9283
Iter19  Ground truth: 0  prediction: 0.0029
Iter20  Ground truth: 1  prediction: 0.9468
Loss: 0.1968  Accuracy: 95.2%
```

把 learning rate 改為原本的 1/10，可以發現結果也很好，這是因為 epoch 的數目夠多，如果發現把 learning rate 改小後結果變差，可以考慮把 epoch 數目變大，如此一來可能也可以得到好的結果，把 learning rate 變小後可以在 error surface 上面做比較細微的移動，如此一來比較有機會可以得到比較低的 loss。

B. Try different numbers of hidden units

左圖為 generate_linear()的結果，右圖為 generate_XOR_easy()的結果

```
Iter92  Ground truth: 0  prediction: 0.0000
Iter93  Ground truth: 1  prediction: 1.0000
Iter94  Ground truth: 0  prediction: 0.0000
Iter95  Ground truth: 0  prediction: 0.0000
Iter96  Ground truth: 0  prediction: 0.0141
Iter97  Ground truth: 1  prediction: 1.0000
Iter98  Ground truth: 1  prediction: 1.0000
Iter99  Ground truth: 1  prediction: 1.0000
Loss: 0.0408  Accuracy: 98.0%
```

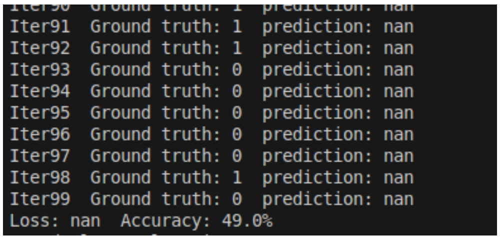
```
Iter13  Ground truth: 0  prediction: 0.0004
Iter14  Ground truth: 1  prediction: 0.9998
Iter15  Ground truth: 0  prediction: 0.0004
Iter16  Ground truth: 1  prediction: 0.9999
Iter17  Ground truth: 0  prediction: 0.0004
Iter18  Ground truth: 1  prediction: 0.9999
Iter19  Ground truth: 0  prediction: 0.0004
Iter20  Ground truth: 1  prediction: 0.9999
Loss: 0.0004  Accuracy: 100.0%
```

這是把 hidden unit 改為 16 個以後的結果，可以發現到結果也很好，這

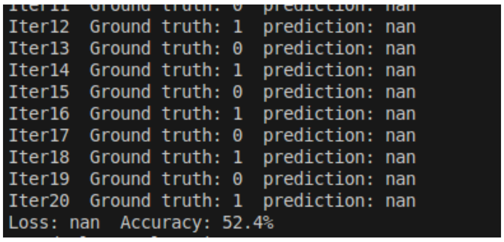
是因為增加 hidden unit 可以讓 error surface 上面的維度變大，也因此在 error surface 上面有夠多的方向可以走，如此一來，便有機會可以得到更好的結果，但缺點是因為變數的數目變多，所以訓練的過程會比較耗時。

C. Try without activation functions

左圖為 generate_linear()的結果，右圖為 generate_XOR_easy()的結果



```
Iter91  Ground truth: 1  prediction: nan
Iter92  Ground truth: 1  prediction: nan
Iter93  Ground truth: 0  prediction: nan
Iter94  Ground truth: 0  prediction: nan
Iter95  Ground truth: 0  prediction: nan
Iter96  Ground truth: 0  prediction: nan
Iter97  Ground truth: 0  prediction: nan
Iter98  Ground truth: 1  prediction: nan
Iter99  Ground truth: 0  prediction: nan
Loss: nan  Accuracy: 49.0%
```



```
Iter12  Ground truth: 1  prediction: nan
Iter13  Ground truth: 0  prediction: nan
Iter14  Ground truth: 1  prediction: nan
Iter15  Ground truth: 0  prediction: nan
Iter16  Ground truth: 1  prediction: nan
Iter17  Ground truth: 0  prediction: nan
Iter18  Ground truth: 1  prediction: nan
Iter19  Ground truth: 0  prediction: nan
Iter20  Ground truth: 1  prediction: nan
Loss: nan  Accuracy: 52.4%
```

可以看到結果是 NaN，這是因為我的 loss function 是用 cross-entropy，如果去掉 activation，則結果會超出[0,1]，造成結果為 NaN

另外，因為去掉 activation function，所以只會做線性的轉換，無法學習複雜的 data 分佈，也會造成最後的預測結果很差。

5. Questions

A. What is the purpose of activation functions?

Activation functions 的作用在於引入非線性的概念，如果沒有 activation function，則無法學習複雜的 data set 的特徵

適合的 activation function 可以讓 back-propagation 時可以順利進行，不會發生上述 NaN 的情況，並且可以提高訓練的穩定性。

B. What might happen if the learning rate is too large or too small?

如果 learning rate 太大，則可能發生跳過 error surface 低谷的情況，這會造成 parameter 的更新來回震盪，但都無法順利到達低谷，另外 learning rate 太大會無法在 error surface 上做微小的移動，這會造成最後的結果往往不是 error surface 的低谷。

如果 learning rate 太小，則會耗費許多訓練的時間，且如果太小的話，可能會陷在一個局部低谷，無法成功到達下一個更低的低谷，反觀如果 learning rate 大一點，則可能越過這個局部低谷，往更深的低谷前進。

C. What is the purpose of weights and biases in a neural network?

Weights 跟 biases 的作用是讓神經網路可以順利地預測 testing set 中的數據，當在訓練時，我們需要對 loss function 做 gradient，並且更新

weights 跟 biases，以期望可以得到更小的 loss，當 training set 跟 testing set 中的資料分布是相似時，我們便可以依照我們訓練時得到的 weights 跟 biases 來預測 testing set 的結果。