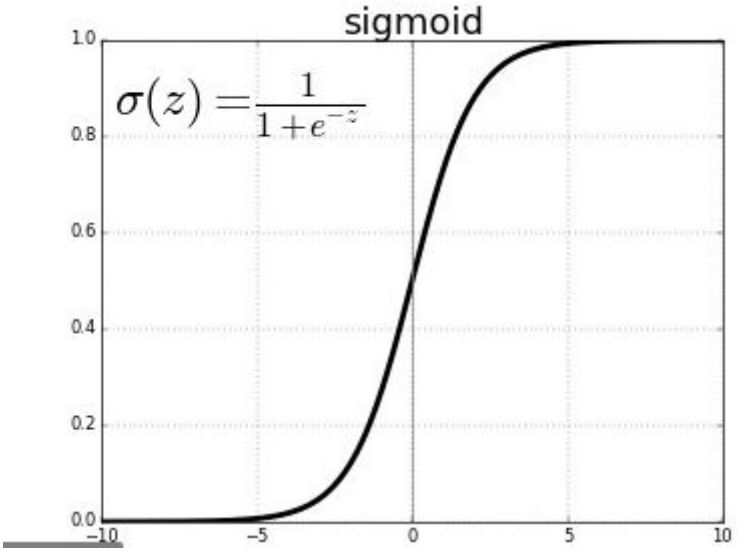
DL HW1 李明峻0856558

Introduction:

用一個簡單的兩層 fully connected neural network 來去學習。在 training 的階段，會將 input 的值 forward 進 neural network，即 inputs 和不同 layers 的 weights 做矩陣運算，最後得出一個 prediction result，我們利用這個 prediction result 去和 Ground truth 計算 loss，再由 loss function 反推回去更新逐層的 weights，重複這步驟去對網路的參數做更新。

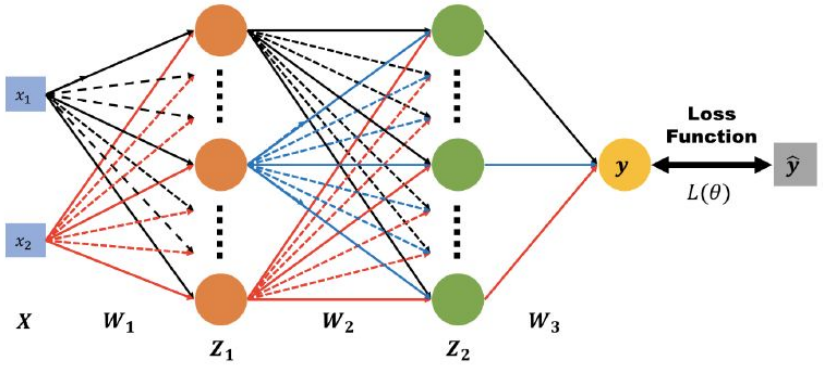
Experiment setups:

A. Sigmoid function:



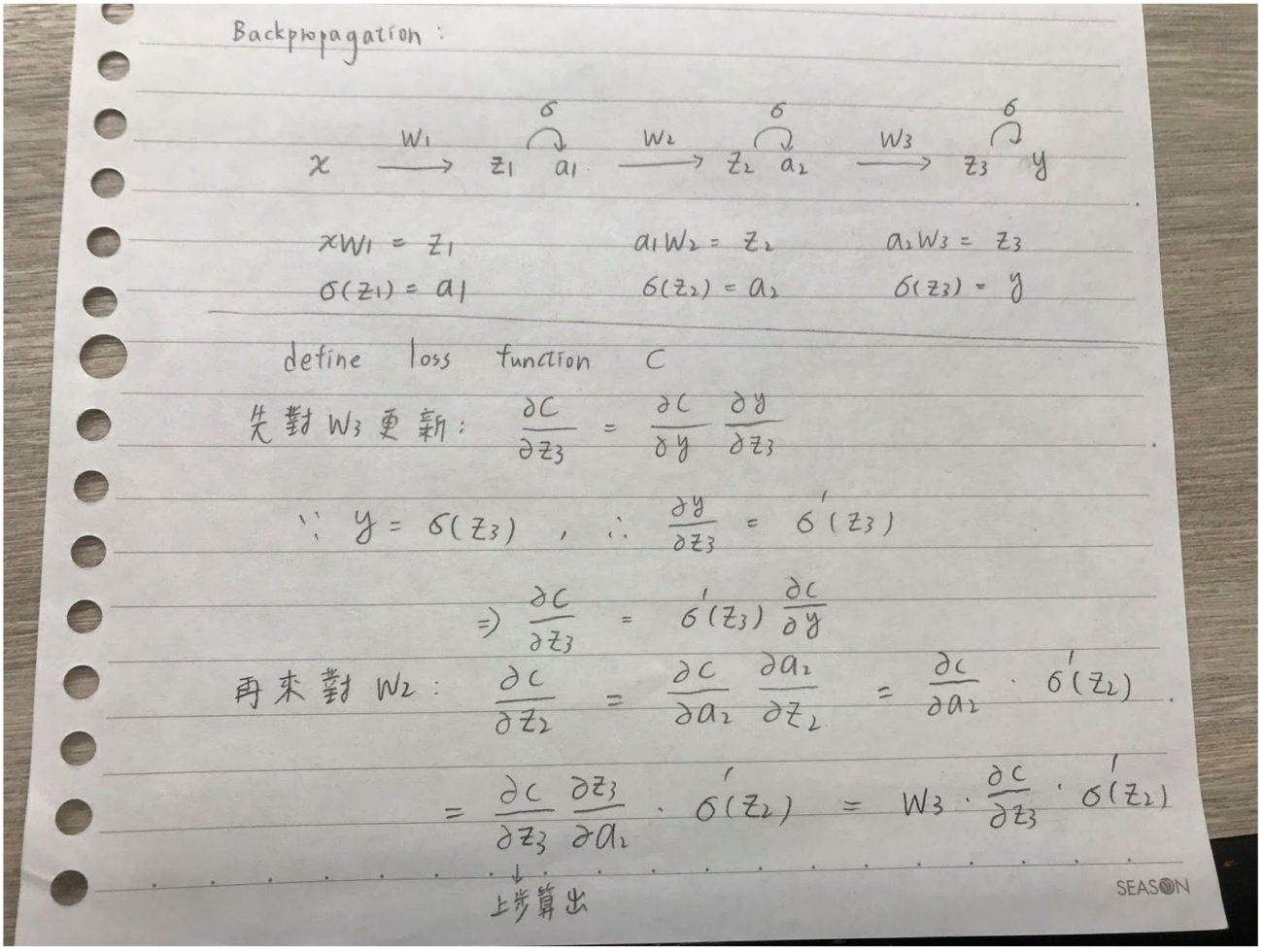
Sigmoid function 就是一個 activation function，將任何變量的值轉換到[0, 1]之間的值，通常被用來當作神經網路的非線性轉換單元。

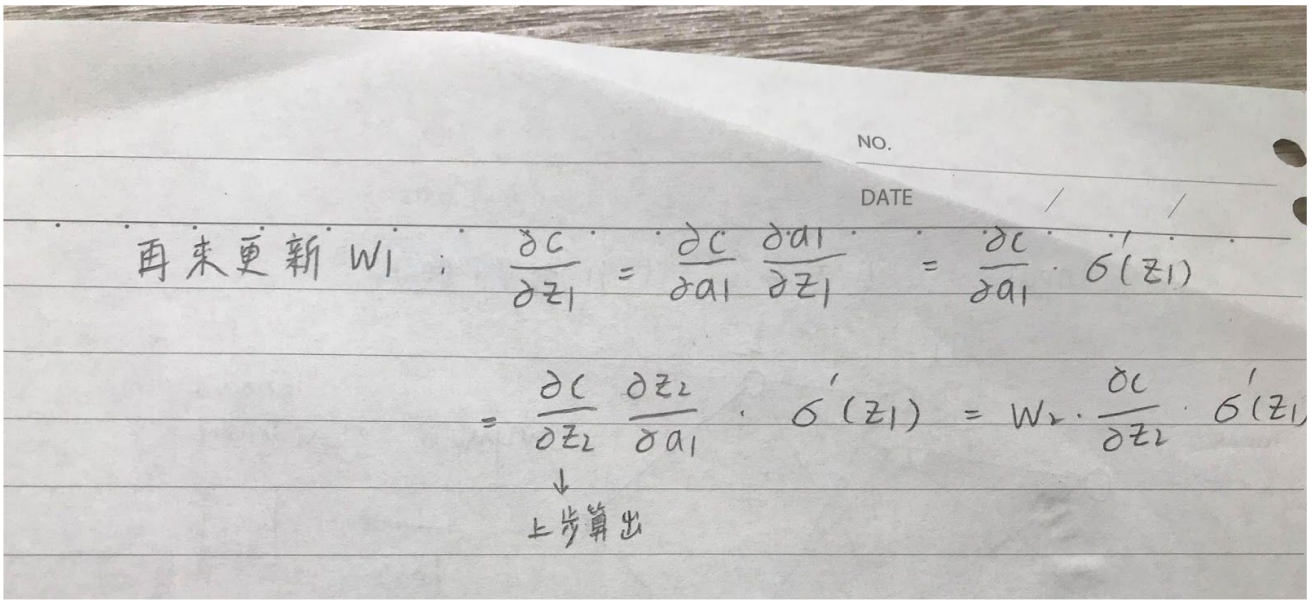
B. Neural Network:



每一層都有 hidden_size 個 neurons：第一層的 weights size 為 [2, hidden_size]，第二層的 weights size 為 [hidden_size, hidden_size]，第三層的 weights size 為 [hidden_size, 1]
Activation function 為 Sigmoid，在 Backpropagation 階段需要 Sigmoid 的微分才能做計算。

C. Backpropagation：
我用手寫推導 Backpropagation





由最後一層推導回去：對於W3，先對 loss function 偏微(利用 chain rule)，可以得到 sigmoid 的微分乘上 loss function 的偏微分，對應程式碼的第二行，再將矩陣存入 delta 中，接著照著同樣步驟算出前幾層的 delta 資訊。

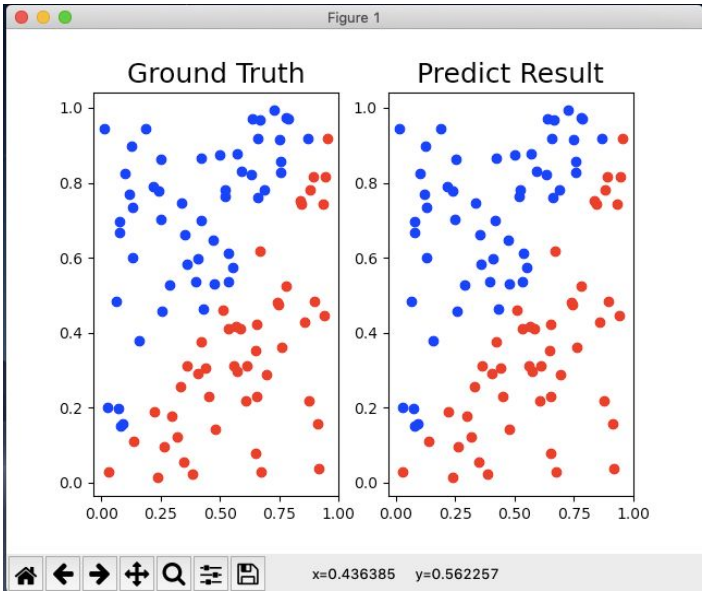
```
delta = []
delta.append(self.der_loss * derivative_sigmoid(self.output))
delta.append(np.dot(delta[-1], self.hidden3_weights.T) * derivative_sigmoid(self.z2))
delta.append(np.dot(delta[-1], self.hidden2_weights.T) * derivative_sigmoid(self.z1))
```

完成後根據特定的 learning rate 去更新每一層的 weights。

Result of testing:

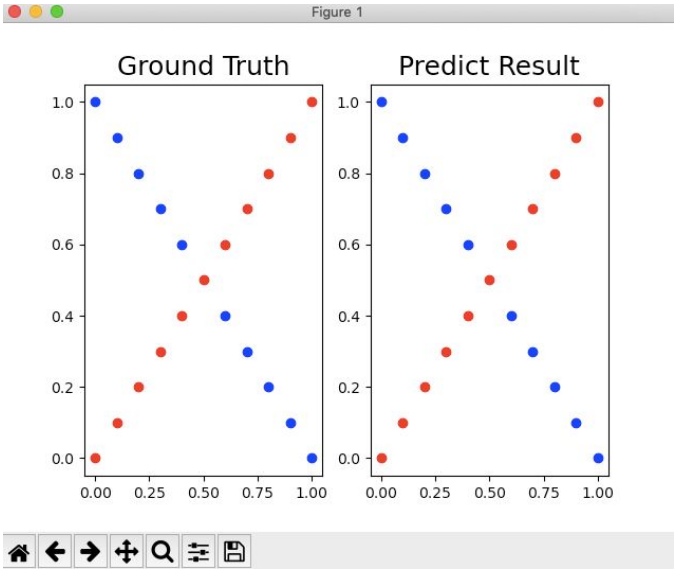
Linear:

```
Epochs 0: accuracy: 86.53% loss: 29.15
Epochs 100: accuracy: 98.96% loss: 1.54
Epochs 200: accuracy: 99.96% loss: 0.04
Epochs 300: accuracy: 99.97% loss: 0.03
Epochs 400: accuracy: 99.98% loss: 0.02
Epochs 500: accuracy: 99.98% loss: 0.02
Epochs 600: accuracy: 99.99% loss: 0.01
Epochs 700: accuracy: 99.99% loss: 0.01
Epochs 800: accuracy: 99.99% loss: 0.01
Epochs 900: accuracy: 99.99% loss: 0.01
Epochs 1000: accuracy: 99.99% loss: 0.01
Epochs 1100: accuracy: 99.99% loss: 0.01
Epochs 1200: accuracy: 99.99% loss: 0.01
Epochs 1300: accuracy: 99.99% loss: 0.01
Epochs 1400: accuracy: 99.99% loss: 0.01
Epochs 1500: accuracy: 99.99% loss: 0.01
Epochs 1600: accuracy: 99.99% loss: 0.01
Epochs 1700: accuracy: 100.00% loss: 0.00
Epochs 1800: accuracy: 100.00% loss: 0.00
Epochs 1900: accuracy: 100.00% loss: 0.00
Training finished! accuracy: 100.00% loss: 0.00
```



XOR:

```
Epochs 0: accuracy: 49.77% loss: 22.70
Epochs 100: accuracy: 99.40% loss: 0.13
Epochs 200: accuracy: 99.70% loss: 0.06
Epochs 300: accuracy: 99.80% loss: 0.04
Epochs 400: accuracy: 99.85% loss: 0.03
Epochs 500: accuracy: 99.88% loss: 0.02
Epochs 600: accuracy: 99.90% loss: 0.02
Epochs 700: accuracy: 99.92% loss: 0.02
Epochs 800: accuracy: 99.93% loss: 0.01
Epochs 900: accuracy: 99.94% loss: 0.01
Epochs 1000: accuracy: 99.94% loss: 0.01
Epochs 1100: accuracy: 99.95% loss: 0.01
Epochs 1200: accuracy: 99.95% loss: 0.01
Epochs 1300: accuracy: 99.96% loss: 0.01
Epochs 1400: accuracy: 99.96% loss: 0.01
Epochs 1500: accuracy: 99.96% loss: 0.01
Epochs 1600: accuracy: 99.97% loss: 0.01
Epochs 1700: accuracy: 99.97% loss: 0.01
Epochs 1800: accuracy: 99.97% loss: 0.01
Epochs 1900: accuracy: 99.97% loss: 0.01
Training finished! accuracy: 99.97% loss: 0.01
```



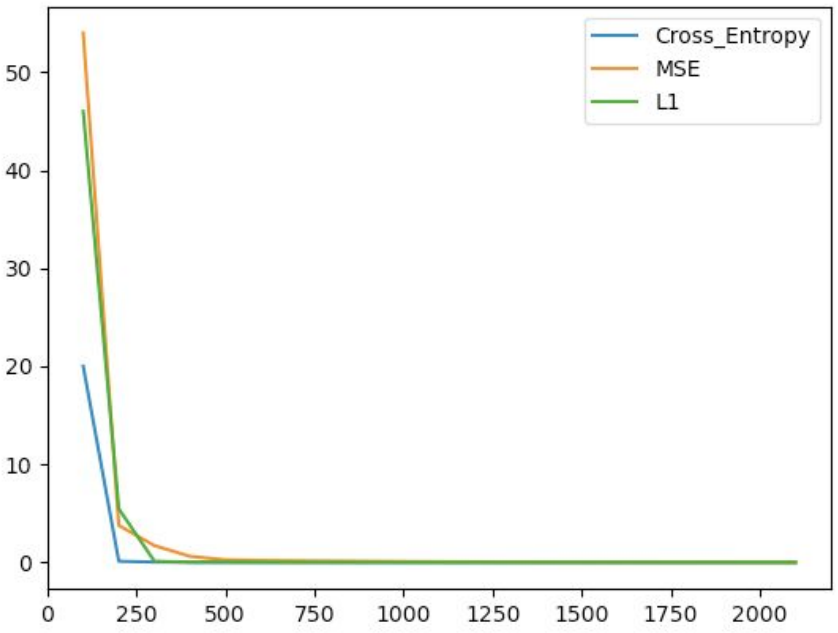
Discussion:

Learning rate: 原先 Learning rate 是 0.2，但是在這兩個較簡單的例子中若將 Learning rate 調過大(0.5以上)都只會讓收斂到最小誤差的速度更快(基本上在第200個Epochs左右準確率就有99%); 若 Learning rate 調到0.1以下，因為參數更新的幅度太小，原先的2000次 training 是不夠收斂到更好的準確率的。

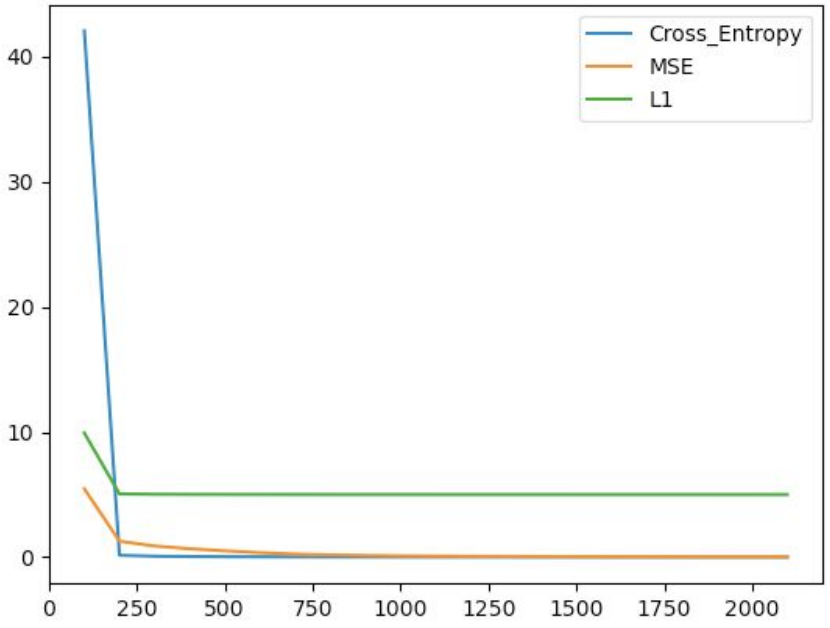
Hidden_size: 試著將每層的 neurons 數目增加，從原本的100個提升到200個，很明顯的訓練速度慢了非常多，但是準確率很快地收斂了(第100個Epochs就有99%的 accuracy)，運算單元複雜化雖然提升了準確率，但是就必須要犧牲掉時間的成本。

Loss function: 比較三種 loss function，MSE,MAE及Cross_Entropy，其中 Cross_Entropy 是收斂最快的。
而對於 XOR 這個 dataset 來說，MAE 並不是個好的 loss function

Linear:


Cross_Entropy
MSE
L1
50
40
30
20
10
0
0
250
500
750
1000
1250
1500
1750
2000

XOR:


Cross_Entropy
MSE
L1
40
30
20
10
0
0
250
500
750
1000
1250
1500
1750
2000