Deep Learning Lab1-Backpropagation Report

智能所 311581006 林子淩

1. Introduction

本次 lab 主題為實作 backward propagation。Backpropagation 是在訓練 deep neural network 不可或缺的步驟，在 backpropagation 的過程中，我們可以通過梯度下降(gradient descent)來由後至前更新 neural network 中每一層 layer 裡面的參數(weight, bias)，使 model output prediction 和 ground truth label 的 error term 逐漸下降。本次 lab 禁止使用常見的 deep learning framework (例如：Pytorch, Tensorflow)，只能使用基本的 numpy 和 python standard libraries 去實作出至少涵括兩層 hidden layer 的 neural network，並建構出 forward pass 和 backward propagation 的計算。本次 lab 的學習目的為讓我們能夠深入瞭解最基本的神經網路架構及訓練參數更新的原理和計算過程。

2. Experiment Setups

A. Sigmoid Functions

Sigmoid function 是一種在 deep learning model 中常用的 activation function，其提供非線性的轉換方式(non-linearity)，讓模型可以有更多的轉換空間去 fit data。在計算上，sigmoid 是一個簡單的映射函數，可以將數字映射到[0,1]的範圍內，為平滑單調遞增函數，整個函數以(0, 0.5)呈現中心對稱。Sigmoid 和 sigmoid 微分的公式如下，圖形如 Figure 1：

$$\text{Sigmoid function: } \sigma(x) = \frac{1}{1+e^{-x}}$$

$$\text{Derivative of sigmoid function: } \sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

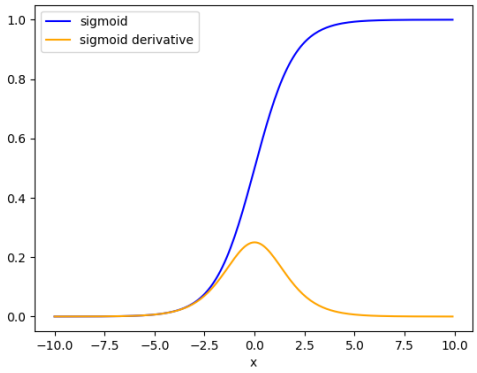


Figure 1　sigmoid 和 sigmoid derivative 圖形

B. Neural Network

在本次 lab 中，我建構了一個 neural network 作為分類器(classifier)來將資料點(data points)分成兩個類別。整個 neural network model 包含一層 input layer、兩層 hidden layer 以及一層 output layer (總共四層)。每一層 layer 後面可以選擇接(sigmoid, ReLu)其中一種作為 activation function 或是選擇不用 activation function (none)，除了 output layer 如果整體選擇 none 作為 activation function，會自動改成 sigmoid。Input layer 每筆資料輸入維度為 2，中間根據不同實驗選擇不同的 hidden units 數量(即 hidden layer 的 node 數量，default 為 50 個)，output layer 輸出維度為 1。當 data point 屬於第一類

時，output 應該輸出 0；屬於第二類則 output 應輸出 1。在實驗中我選擇將 threshold 設成 0.5 (考慮到使用 default activation function 為 sigmoid，而 sigmoid 的映射範圍介於 0 和 1 之間)，因此 output layer 輸出的 prediction value 如果小於/等於 0.5 會被分作第一類，大於 0.5 則分作第二類。

C. Backpropagation

在執行 backpropagation 步驟，首先需要先計算 Loss value 來量化 model prediction 和 ground truth label 之間的差異，再利用 gradient descent 來循環更新 model weight 和 bias (由最後一層 layer 開始向前更新)。本實驗選擇使用的 loss function 為 Mean Square Error (MSE)，目標為最小化 model prediction value 和 ground truth 之間的差，其公式如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

公式中，n為 training data 筆數，y_i和$\hat{y}_i$分別代表第i筆資料的 ground truth label 和 prediction value。

本次 lab 中使用的梯度下降法為 Stochastic gradient descent (SGD)，SGD 的精神為修改 model 參數θ來達到降低 loss value $C(\theta)$的目標($C(\theta)$為 loss 的通稱，在本次實驗中為 MSE)。通過計算 loss $C(\theta)$對參數θ的微分，可以得到梯度(gradient)，梯度的物理意義為參數θ沿著該方向變化會讓 loss $C(\theta)$變化最快、變化量最大。當參數θ沿著 gradient 的正向方向移動，loss $C(\theta)$會上升；反之，當參數θ沿著 gradient 的負向方向移動，loss $C(\theta)$會朝著最有效率的方向下降，即朝著 local minima 移動，達到 minimize loss function 的訓練目的。SGD 參數更新法相當直觀，然而若學習率(learning rate，可以看作更新參數的幅度)太大，這種方法容易造成參數θ在 local minima 附近來回震盪，沒有辦法順利地降低 loss $C(\theta)$。實現 SGD 的方法為計算 loss 的梯度$\nabla C(\theta)$，根據微積分 chain-rule 可得到：

$$\nabla C(\theta) = \frac{\partial C(\theta)}{\partial w_i} = \frac{\partial z^l}{\partial w_i}\frac{\partial C(\theta)}{\partial z^l}$$

公式中，w_i為 layer l的 weight，z^l為 layer l的 output。

$\frac{\partial z^l}{\partial w_i}$：根據 forward pass， $z^l = input^l * w_i + b$。因此這一項偏微分的結果其實就是 layer l的 input。

$\frac{\partial C(\theta)}{\partial z^l}$：將每一層 layer l的後面這項偏微分簡寫為$\delta^l$，即 backward propagation 由後往前計算至 layer l 前更新的 gradient。output layer 的δ^{output}公式如下：

$$\delta^{output} = \sigma'(z^{output-1})\frac{\partial C(\theta)}{\partial y}$$

$\frac{\partial C(\theta)}{\partial y}$為 loss derivative (loss value 對 model prediction y 的微分)，$\sigma'(z^{output-1})$則為 sigmoid derivative，其輸入$z^{output-1}$為 output layer 前一層 layer 的輸出。其他 layer 的δ^l公式如下：

$$\delta^l = \sigma'(z^l)\delta^{l+1}W_{l+1}$$

δ^{l+1}可看作 backward propagation 前一個節點輸出的 gradient，因為順序為由後向前更新，因此需要將此 gradient δ^{l+1}矩陣相乘上通往該層(即 layer $l+1$)的 weight matrix W_{l+1}，再通過 activation function derivative 才可得到當前一層(layer l)的δ^l。(注意此項為總梯度$\nabla C(\theta)$計算的後項偏微分結果，需要再和前項偏微分結果做 dot product 才能得到當前 layer 需要更新的 gradient。另外必須考量矩陣形狀，因此在算$\nabla C(\theta)$時，$\frac{\partial z^l}{\partial w_i}$必須做 transpose；同理，在計算$\delta^l$時，$W_{l+1}$必須 transpose。

3. Results of your testing

A. Screenshot and Comparison Figure

在 default setting (hidden units=50, activation function=sigmoid, learning rate=1e-1)的設定之下，實驗的 loss 和 accuracy 結果如 Table 1 所示。可以看出隨著 epoch 升高，loss 會逐漸下降，代表模型確實有逐漸學習到如何分類 data，而 accuracy 則上升直到等於 1 (完全分類正確)時停止訓練。比較 Linear data 與 XOR data 的訓練過程可以發現，XOR data 需要的訓練 epoch 較少，比較容易訓練。

Training loss and accuracy	
Linear data	**XOR data**
exp_iteration 0 epoch 0, loss 0.2495, acc 0.5800 epoch 5000, loss 0.2436, acc 0.5800 epoch 10000, loss 0.2436, acc 0.5800 epoch 15000, loss 0.2436, acc 0.5800 epoch 20000, loss 0.2436, acc 0.5800 epoch 25000, loss 0.2436, acc 0.5800 epoch 30000, loss 0.2436, acc 0.5800 epoch 35000, loss 0.2435, acc 0.5800 epoch 40000, loss 0.2431, acc 0.5800 epoch 45000, loss 0.1351, acc 0.9300 epoch 46077, loss 0.0512, acc 1.0000	exp_iteration 0 epoch 0, loss 0.4303, acc 0.4762 epoch 5000, loss 0.2486, acc 0.5238 epoch 10000, loss 0.2459, acc 0.5714 epoch 15000, loss 0.2350, acc 0.6667 epoch 20000, loss 0.2041, acc 0.8095 epoch 25000, loss 0.1958, acc 0.8095 epoch 29491, loss 0.1438, acc 1.0000

Table 1　Training loss 和 accuracy 截圖

Table 2 展示了 Linear 和 XOR dataset 中每一個 data point 對應的 model prediction 結果。截圖是還沒有經過 threshold 分類的結果，因為本次實驗中分類 threshold 設定為 0.5，因此 model prediction 小於等於 0.5 的會被分到第一類，大於 0.5 則分到第二類，以此得到最終的 label。

Model predictions	
Linear data	**XOR data**
[[0.8675135 0.87453227 0.47357851 0.75233983] [0.7265744 0.84189297 0.97832467 0.97565267] [0.72512699 0.63152609 0.74147521 0.90949069] [0.54639107 0.48826707 0.98156181 0.88473847] [0.09284852 0.12077756 0.72365807 0.98373339] [0.21433675 0.95225932 0.89627791 0.03874752] [0.32774774 0.97744831 0.62358922 0.09302397] [0.09674054 0.68225751 0.87836508 0.18724593] [0.94840073 0.32481026 0.12698657 0.02007763] [0.97796495 0.23956921 0.04535432 0.49017097] [0.20005087 0.10627042 0.97273069 0.95738708] [0.46968113 0.98942787 0.70907497 0.30702413] [0.14554586 0.97225635 0.78920337 0.05770487] [0.17801095 0.99243104 0.86427676 0.49999432] [0.22053524 0.98020524 0.74476288 0.04513179] [0.56573392 0.81285124 0.15222037 0.85556308] [0.74975109 0.90806487 0.05663709 0.04877199] [0.09728782 0.3585952 0.10049738 0.63964175] [0.64224705 0.94740762 0.05540817 0.99179985] [0.12962897 0.9482769 0.91404304 0.34573065] [0.83682503 0.7890707 0.35763848 0.92727598] [0.47820835 0.97774197 0.26838855 0.77700246] [0.10635215 0.87426376 0.09958528 0.02881265] [0.9599164 0.93597513 0.91586137 0.28145268] [0.57902418 0.55349801 0.8153425 0.881348]]	[[0.22446953] [0.69336926] [0.22541739] [0.64851706] [0.26508577] [0.59915436] [0.34456626] [0.55293609] [0.43821063] [0.51760754] [0.49890648] [0.49818798] [0.50041626] [0.43655624] [0.52466991] [0.33556638] [0.57357514] [0.22962348] [0.64648388] [0.14559438] [0.73600301]]

Table 2　Linear 和 XOR datasets 中每一個 data point 對應的 model predictions

Table 3 為 ground truth 與 predict result 的分類結果視覺化。圖中的 predict result 為 model 訓練到 accuracy=1 的分類結果，因為本次實驗沒有區分 training set 和 test set，預測資料與訓練資料一樣，

所以可以訓練到跟 ground truth 完全一致。

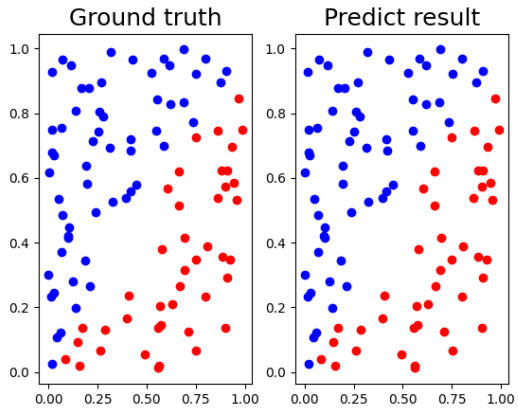
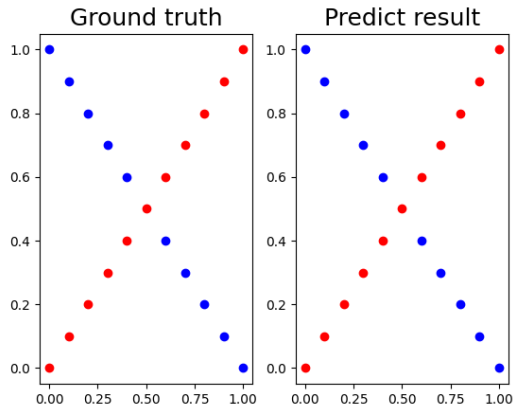
Comparison between Ground truth and Predict result	
Linear data	**XOR data**
	

Table 3　比較 Ground truth 與 Prediction result

B. Show the Accuracy of your Prediction

在 Table 2 中以每 5000 個 epoch 一次的頻率以文字顯示 training accuracy，可以看出隨著 loss 降低，accuracy 逐漸升高至 1，模型成功訓練到能夠完全正確分類。Table 4 進一步將 accuracy 紀錄視覺化，通過觀察 training accuracy curve 可以看出 Linear data 在訓練後期 accuracy curve 急遽上升(找到接近 local minima 的參數組合)。XOR data 則是因為資料點較少所以 accuracy curve 比較崎嶇(每多分類對一個資料點 accuracy 就會上升一階)。同時也猜測可能是因為實驗選擇的 learning rate 1e-1 比較大，所以兩者訓練過程中可能一直在 local minima 附近來回震盪，直到剛好找到 local minima 後 accuracy curve 才上升至 1。

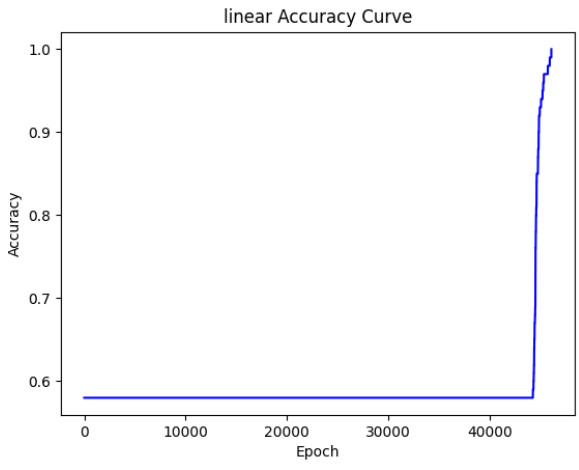
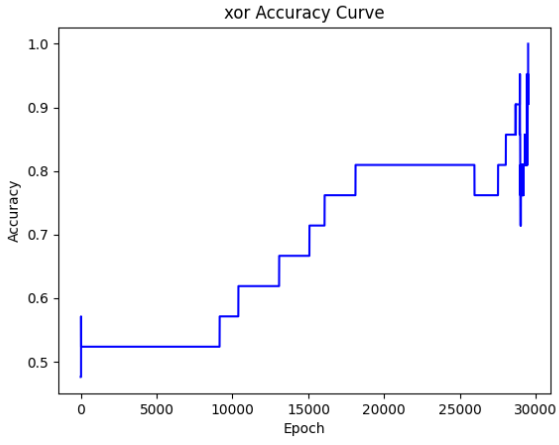
Accuracy Learning Curve	
Linear data	**XOR data**
	

Table 4　Training accuracy curve

C. Learning Curve

Table 4 和 Table 5 分別為 Accuracy Learning Curve 和 Loss Learning Curve，通過比對兩者可以發現對應關係，當 loss 急遽下降代表模型找到接近 local minima 的參數組合，此時的 accuracy 對應也會急遽上升(分類正確性提高)。而 loss 在逐漸下降後突然上升可能是因為 learning rate 較大所以震盪到距

離 local minima 比較遠的位置，此時對應的 accuracy 也會下降。但整體來說，大致上呈現隨著 epoch 升高，loss 下降，accuracy 上升的趨勢，代表模型隨著訓練次數上升逐漸學會如何正確分類。

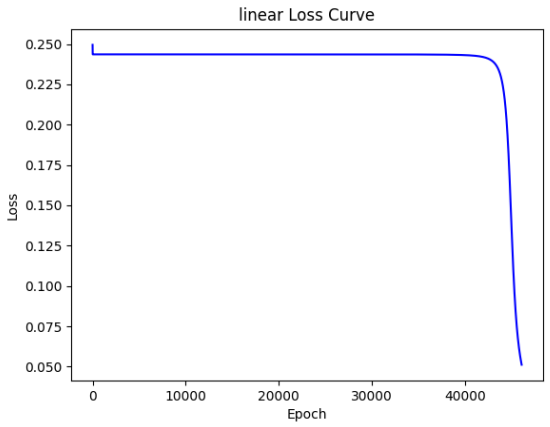
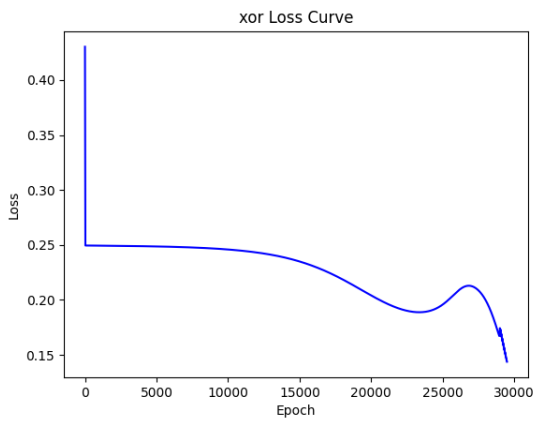
Loss Learning Curve	
Linear data	**XOR data**
	

Table 5　Training loss curve

4. Discussion

第四部份探討實驗數據與其餘額外實驗的結果，以表格記錄不同參數實驗 (調整 Learning Rate, Hidden Units, Activation Function) 之下，Linear data 與 XOR data 訓練的收斂情況和 Accuracy 表現。本次實驗對每一個參數組合都進行了十次實驗，並回報十次實驗的平均結果。其中，Stop Epoch 代表十次訓練平均在第幾個 epoch 第一次達到 accuracy 為 1，即完全分類正確的情況。# of Converge Failure 代表在十次實驗中，有幾次沒有在 100000 個 epoch 內達到 accuracy 為 1，即沒有成功收斂(此時的 Stop Epoch 會被記錄成 100000+)。Best Acc 代表十次實驗中達到過最好的 accuracy 表現(單次實驗結果，非平均)。Avg of Best Acc 則代表十次實驗 Best Acc 的平均。

A. Try different learning rates

Table 6 為使用不同 learning rates，固定 hidden units=50, activation function=Sigmoid 的實驗結果。從表中可以看出，當 learning rate 為 1e-1 時，Linear 和 XOR 兩種資料的訓練都會收斂最快(Stop Epoch 最小)。而當 learning rate 減半至 5e-2 時，兩個資料集仍然能夠在 100000 個 epochs 內收斂，但收斂速度明顯較慢，需要的訓練 epoch 數量上升。當 learning rate 再降低至 1e-2 和 5e-3 時，因為更新梯度的幅度變小，需要超過 100000 個訓練 epochs 才能達到收斂，學習到如何正確分類所有 data points，只訓練 100000 個 epochs 不足以讓模型的 accuracy 上升至 1(灰色標示部分)。整體數據符合當 learning rate 越小，模型會需要更多 epoch 以達到收斂的邏輯。

Param.			Linear/XOR			
Learning Rate	**Hidden Units**	**Activation Function**	**Stop Epoch**	**# of Converge Failure**	**Best Acc**	**Avg of Best Acc**
1e-1	50	Sigmoid	44344/24014	0/0	1/1	1/1
5e-2	50	Sigmoid	88683/62027	0/0	1/1	1/1
1e-2	50	Sigmoid	100000+/100000+	10/10	0.58/0.86	0.58/0.65
5e-3	50	Sigmoid	100000+/100000+	10/10	0.58/0.52	0.58/0.52

Table 6　Experiment of different learning rates

B. Try different numbers of hidden units

Table 7 為使用不同 hidden units，固定 learning rate=1e-1, activation function=Sigmoid 的實驗結果。從表中可以看出，對於 Linear 和 XOR 兩種資料來說，當 hidden units 越多(上限 100 為例)，模型參數越多，其 fit data 的能力越好，需要的 training epoch 也較少。整體而言，hidden units 在 50 以上的模型皆可完美分類 Linear 和 XOR 的資料，收斂達到 accuracy=1 的表現。當 hidden units 在 40 以下，模型會不夠複雜，沒辦法保證完美分類 XOR 資料點，在十次實驗中會有 1~2 次無法在 100000 個 training epochs 內成功收斂(灰色標示部分)。另外，Stop Epoch 雖整體隨 hidden units 上升而逐漸下降的趨勢，但仍有幾個實驗不符合此規律(例如：hidden units=50 時，XOR data 的 Stop Epoch 為 24014，少於 hidden units=60 時 XOR data 的 Stop Epoch 26428)，推測當 hidden units 上升時模型雖參數增加可以 fit 更複雜的資料點，但穩定性會稍微降低。

Param.			Linear/XOR			
Learning Rate	Hidden Units	Activation Function	Stop Epoch	# of Converge Failure	Best Acc	Avg of Best Acc
1e-1	10	Sigmoid	57100/59377	0/2	1/1	1/0.99
1e-1	20	Sigmoid	53253/49556	0/2	1/1	1/0.99
1e-1	30	Sigmoid	48832/47748	0/2	1/1	1/0.99
1e-1	40	Sigmoid	44899/39934	0/1	1/1	1/0.99
1e-1	50	Sigmoid	44344/24014	0/0	1/1	1/1
1e-1	60	Sigmoid	39880/26428	0/0	1/1	1/1
1e-1	70	Sigmoid	37824/20959	0/0	1/1	1/1
1e-1	80	Sigmoid	35516/20475	0/0	1/1	1/1
1e-1	90	Sigmoid	33543/15276	0/0	1/1	1/1
1e-1	100	Sigmoid	32807/16204	0/0	1/1	1/1

Table 7　Experiment of different numbers of hidden units

C. Try without activation functions

Table 8 為嘗試不加入 activation function 的實驗結果(Activation Function=None)，實驗固定 learning rate=1e-1、hidden units=50，與使用 Sigmoid 作為 activation function 做比較。由表中可以看出，當不使用 activation function 時，在針對 XOR data 的十次實驗皆無法在 100000 個 epochs 內收斂(灰色標記部分)，最高 accuracy 只達到 0.52。因為分類類別只有兩類，avg accuracy=0.58 只比隨機猜測 label 的 accuracy 期望值略高。分類效果差的原因在於如果不加入 activation function，模型會缺乏非線性關係(non-linearity)，單靠線性組合很難去 fit XOR data，因為 XOR 屬於 non-linear data distribution。然而此次的實驗設定為模型 output layer 必須接上一個 activation function (若設定為 None 則 output layer 自動接 Sigmoid)，因此即使設定 activation function 為 None，在最後輸出時仍會經過 Sigmoid function 映射到[0,1]範圍，所以 accuracy 至少有 0.58 的水準。

Param.			Linear/XOR			
Learning Rate	Hidden Units	Activation Function	Stop Epoch	# of Converge Failure	Best Acc	Avg of Best Acc
1e-1	50	Sigmoid	44344/24014	0/0	1/1	1/1
1e-1	50	None	2991/100000+	0/10	1/0.76	1/0.58

Table 8　Experiment of activation functions (Sigmoid and None)

5. Extra

B. Implement different activation functions

在加分項目的部分，本實驗加入了 ReLU 作為 Sigmoid 以外的 activation function 選擇。ReLU 同樣作為一種 deep learning model 常用的 activation function，可以提供模型非線性組合能力(non-linearity)，ReLU 和 ReLU 微分的公式如下，圖形如 Figure 2：

$$\text{ReLU function: } \mathrm{ReLU}(x) = \max(0, x)$$

$$\text{Derivative of ReLU function: } ReLU'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

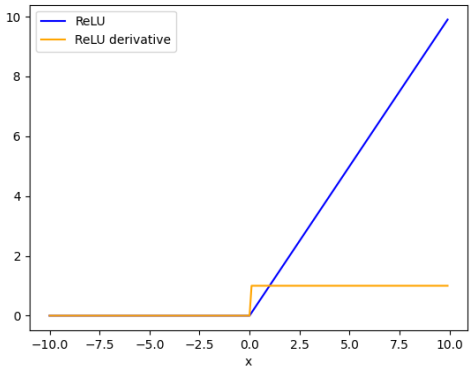


Figure 2　ReLU 和 ReLU derivative 圖形

Table 9 為比較 Sigmoid, ReLU, None 三種 activation function choices 的實驗數據，實驗參數固定 learning rate=1e-1、hidden units=50。從表中可以觀察到使用 ReLU 作為 activation function 的話，在 Linear 和 XOR data 的十次實驗中皆無法收斂。這可能是因為本次實驗 output 只有一個維度，分類以 0 或 1 作為 target label，而使用 ReLU 無法將 output value 映射到[0,1]範圍之內，因此模型在 output value 離 0,1 很遠的情況下不容易收斂。從數據中可以很明顯看出使用 Sigmoid 作為 activation function 的訓練效果較 ReLU 好(所需 epoch 較少，accuracy 較高)，同樣也是因為實驗設定 label 介於[0,1]之間，使用 Sigmoid 作為 mapping function 更合適。None 雖然只在最後 output layer 後接上 Sigmoid，在其餘 layer 皆沒有加入 non-linearity 導致在針對 non-linear distribution XOR data 的訓練效果差於 Sigmoid，但其在 Linear 和 XOR data 的訓練效果仍比 ReLU 好，這也是因為在最後輸出時有將 output value 控制在[0,1]範圍內的緣故。

Param.			Linear/XOR			
Learning Rate	Hidden Units	Activation Function	Stop Epoch	# of Converge Failure	Best Acc	Avg of Best Acc
1e-1	50	Sigmoid	44344/24014	0/0	1/1	1/1
1e-1	50	ReLU	100000+/100000+	10/10	0.42/0.52	0.42/0.52
1e-1	50	None	2991/100000+	0/10	1/0.76	1/0.58

Table 9　Experiment of activation functions (Sigmoid, ReLU and None)