Deep Learning and Practice

#Lab01 Back Propagation　309505002 鄭紹文

1. Introduction

深度學習是機器學習的一大分支，如 fig.1 所示，其模仿人類的大腦神經網絡的運作方式，常見的深度學習架構，有多層感知器(Multilayer Perceptron)、深度神經網路 DNN (Deep Neural Network)、卷積神經網路 CNN(Convolutional Neural Network)、遞迴神經網路 RNN (Recurrent Neural Network)。

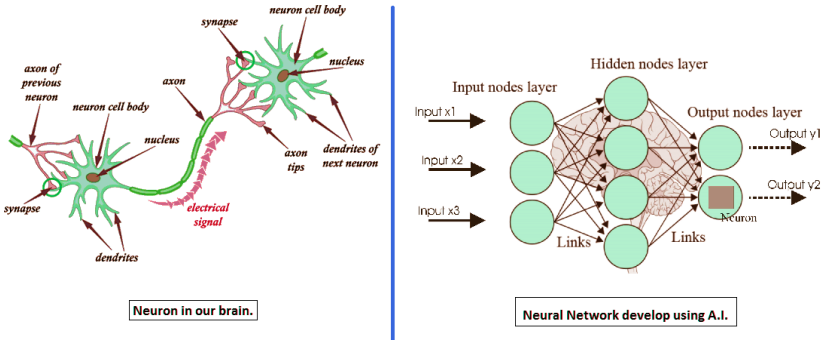


fig. 1

在各類網絡中，weight 的更新是不可或缺的，而這次 lab 的重點：「誤差反向傳播法」(Back Propagation)便是能以良好效率計算出權重參數梯度的方法。

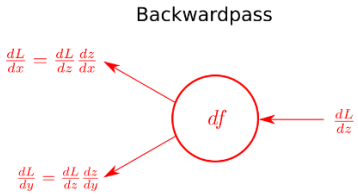


fig. 2

此次 lab 要刻出一個含有 2 層隱藏層的網絡，input data 分別是隨機產生的 linear dataset (100 組)和 XOR dataset。

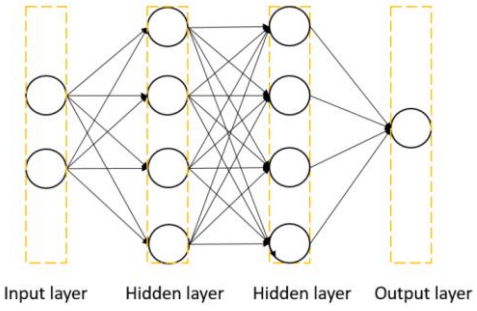


fig. 3

2. Experiment setups

A. Sigmoid functions

Sigmoid function 是神經網路常用的活化函數之一，用於將輸入值轉換為 0~1 (0%~100%)的值，可當作機率值來使用，同時亦可增加非線性，增加非線性讀原因在於若僅用線性，有時會限制，如 data 中簡單的 XOR 分類問題，若用單純的線性 model 會解不出來，而加上非線性可想像成把資料先做對折再切割分類，以此來完成無法利用一條線切割資料分部的問題。

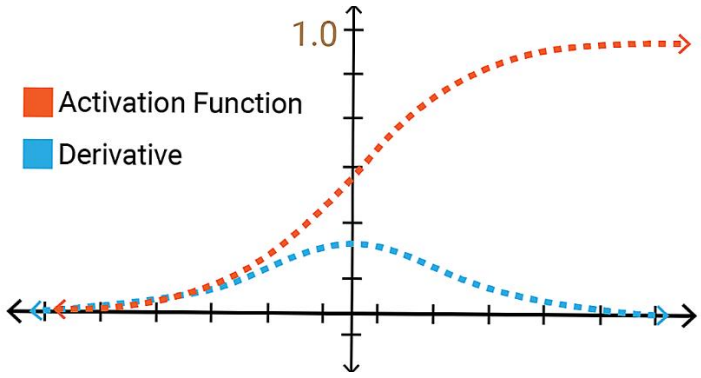


fig. 4

Sigmoid function 常見形式：$\boldsymbol{y = \frac{1}{1+e^{-(ax+b)}}}$

fig.4 中紅線部分即為 Sigmoid 函數，呈現 S 型，亦稱為 S function，同時可從中發現幾項性質：

- 其為函數值介於 0~1 之間的單調遞增函數。
- X 趨近於-∞時，函數值趨近於 0，反之趨近於 1。
- X=0 時，函數值為 0.5 且於(0, 0.5)處會有轉折。

(將 x, -x 分別帶入後相加後即可找到中間點)

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

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

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

$$= -(1+e^{-x})(1+e^{-x})(-e^{-x})$$

$$= \sigma(x)(1-\sigma(x))$$

fig. 5

Sigmoid function 的**微分**為 fig.4 中藍線部分以及 fig.5 紅框處。

B. Neural Network

Neural Network 是模仿生物神經網路的結構和功能的數學模型或計算模型，也有人稱為 ANN(Artificial Neural Network)。神經元(Neuron)之間互相連結，由外部神經元接收信號，再層層傳導至其他神經元，最後作出反應的過程經過抽象化後，得到 fig.6 中類似結構，Input Layer 即為接收信號的神經元，Hidden Layer 即為隱藏層，Output Layer 則是做出反應的輸出層，各神經元傳導的力量大小，在神經網絡中抽象化成「權重」(Weight)，也就是模型要求解的參數，當更新完成後，即可輸入信號，透過一層一層的傳導，推斷出最終結果。除了 Neural unit 和神經元連接的權重代表的結構（Architecture）外，用來定義神經元如何根據其他神經元的活動來改變自己的 activation function 以及指定網路中的權重如何隨著時間推進而調整 Learning Rate 亦是極為重要的一部分。

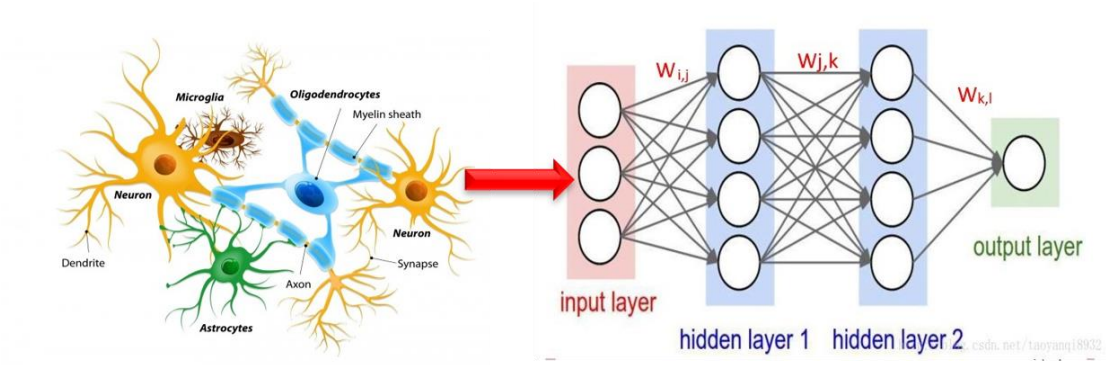


fig. 6

C. Backpropagation

Backpropagation是一種與最優化方法（如梯度下降法）結合使用，用來訓練人工神經網絡的常見方法，其對網絡中所有權重計算損失函數的梯度，而梯度會反饋給最優化方法，用來更新權值以最小化損失函數。

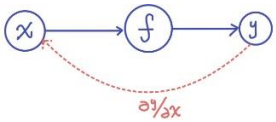


fig. 7

fig.7 為只有一個神經元時的 backpropagation 的反向傳播示意圖，其只能應對相對單純的線性問題；但藉由多個神經元進行階層化，如 fig.8，便能應對複雜的非線性問題。

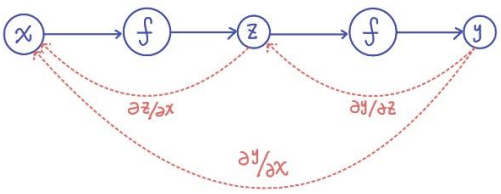


fig. 8

backpropagation 中 chain rule 是一個很重要的環節。

$$\frac{\partial L(\theta)}{\partial w_1} = \frac{\partial y}{\partial w_1}\frac{\partial L(\theta)}{\partial y}$$
$$= \frac{\partial x''}{\partial w_1}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}$$
$$= \frac{\partial z}{\partial w_1}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}$$
$$= \frac{\partial x'}{\partial w_1}\frac{\partial z}{\partial x'}\frac{\partial x''}{\partial z}\frac{\partial y}{\partial x''}\frac{\partial L(\theta)}{\partial y}$$

fig. 9

fig.9 顯示 weight w1 如何影響 loss 值，並利用該值乘上 learning rate 來更新 w1 參數，使 loss 值越變越小。

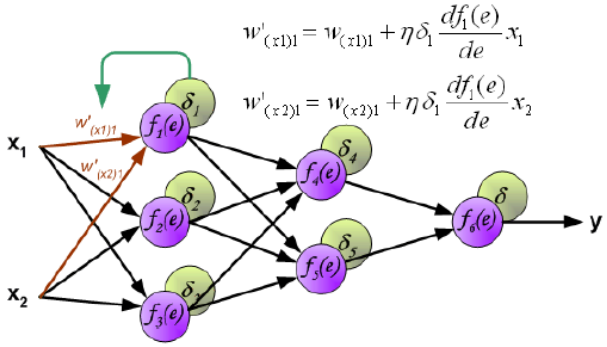


fig. 10

fig.10 為整體架構樣貌，以及其權重如何進行更新，較為細部部分可見 fig.11。

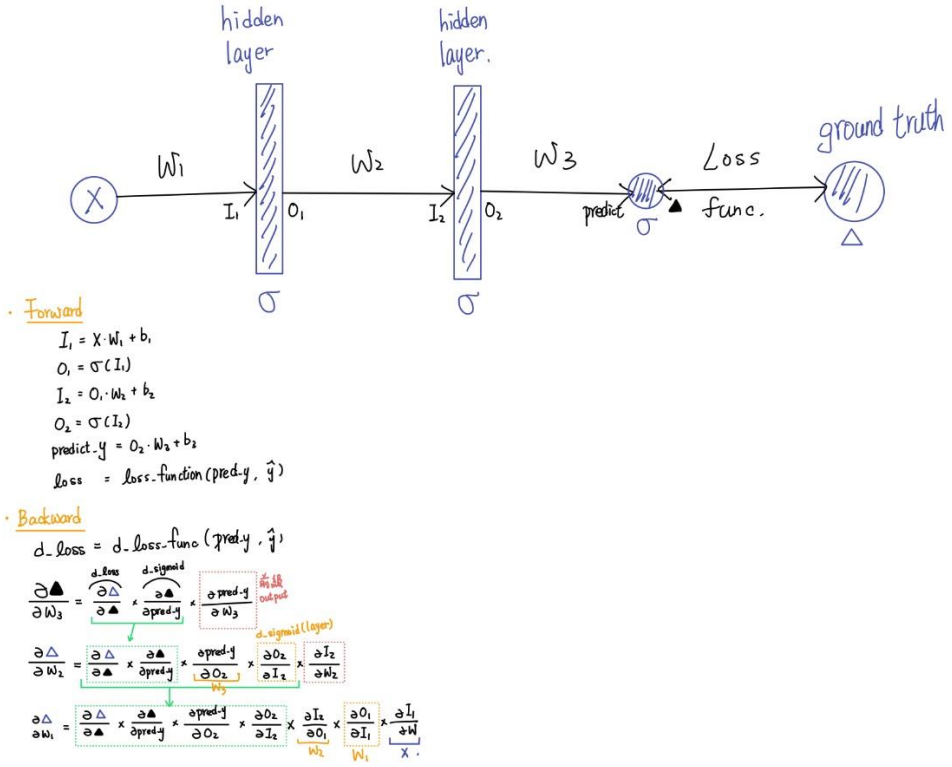


fig. 11

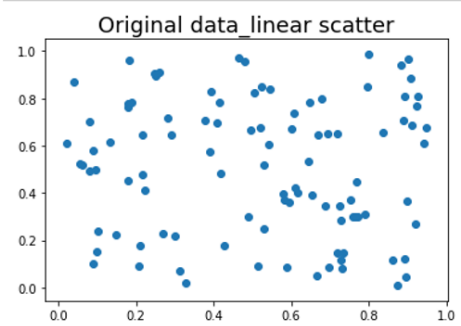
3. Results of your testing

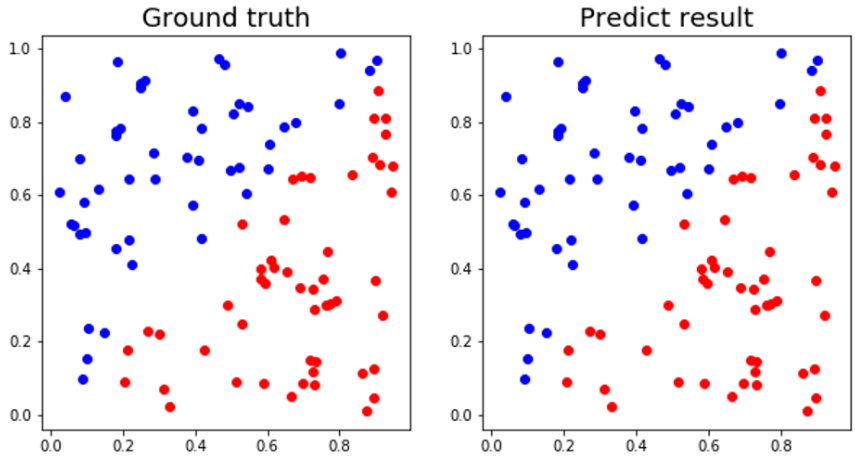
A. Screenshot and comparison figure

- For data type one：**Linear datasets**

```
epoch 10000 Loss: 1.4783568846517297e-05
epoch 15000 Loss: 8.895744082853413e-06
epoch 20000 Loss: 6.2462125512593106e-06
epoch 25000 Loss: 4.763673286753503e-06
epoch 30000 Loss: 3.824956784116845e-06
epoch 35000 Loss: 3.1810573196805137e-06
epoch 40000 Loss: 2.713882963605246e-06
epoch 45000 Loss: 2.360539467041129e-06
epoch 50000 Loss: 2.0845879377594596e-06
epoch 55000 Loss: 1.8635201890220776e-06
epoch 60000 Loss: 1.6827134612324597e-06
epoch 65000 Loss: 1.5322721585088239e-06
epoch 70000 Loss: 1.4052700924326744e-06
epoch 75000 Loss: 1.2967201650007672e-06
epoch 80000 Loss: 1.2029437464434612e-06
epoch 85000 Loss: 1.121170269345594e-06
epoch 90000 Loss: 1.0492748533369948e-06
epoch 95000 Loss: 9.856016028551594e-07
epoch 100000 Loss: 9.288417089377365e-07
```

```
**-----Preduction Result-----**
[[8.01623989e-08 1.13711433e-09 1.89564746e-10 9.99999994e-01
  1.41668620e-09]
 [9.99999999e-01 9.99999999e-01 1.03421634e-10 4.09696597e-10
  9.99984309e-01]
 [1.08622441e-10 1.11432757e-10 2.29911038e-10 9.99999992e-01
  9.99999999e-01]
 [9.99999999e-01 9.98260830e-01 9.99999997e-01 9.99999999e-01
  9.99999257e-01]
 [5.51052534e-09 8.83217362e-11 1.56963099e-10 9.99999999e-01
  2.88489140e-05]
 [9.99999998e-01 1.32937581e-08 9.99999881e-01 9.99999999e-01
  9.99999999e-01]
 [9.99999931e-01 7.76613733e-11 6.25366730e-09 9.96000688e-01
  5.26685771e-10]
 [9.99999999e-01 9.99999992e-01 3.82630590e-07 1.12388543e-10
  2.11259599e-10]
 [9.99999999e-01 5.27587609e-09 2.27716762e-09 9.99994983e-01
  5.28365757e-10]
 [4.69353651e-03 7.65896804e-08 1.67987756e-06 9.99999099e-01
  2.43882493e-09]
 [9.99999996e-01 4.54359556e-09 2.09381460e-09 9.99386168e-01
  9.99999998e-01]
 [4.13441399e-04 9.99999998e-01 9.99999995e-01 9.99999998e-01
  1.25228352e-10]
 [1.12872876e-10 9.99552348e-01 1.93223442e-10 1.03009132e-09
  9.99999998e-01]
 [9.99997432e-01 1.60243859e-09 1.80263744e-10 5.81067488e-07
  9.99999992e-01]
 [7.48360805e-10 9.99999999e-01 9.99999966e-01 9.99999998e-01
  3.09796945e-10]
 [5.63345302e-04 4.78190137e-03 9.99784745e-01 7.75563728e-10
  9.99999999e-01]
 [1.52898626e-10 9.99999999e-01 2.59981467e-09 9.99738562e-01
  9.99999896e-01]
 [4.68269031e-09 9.99999756e-01 7.61764829e-11 8.62491811e-11
  9.99999893e-01]
 [2.41802654e-06 9.99999999e-01 9.94740165e-01 3.12837145e-10
  9.99999997e-01]
 [1.84441252e-10 9.99999994e-01 9.99999998e-01 3.49220249e-04
  1.19464543e-10]]
**---------------------------------**
```

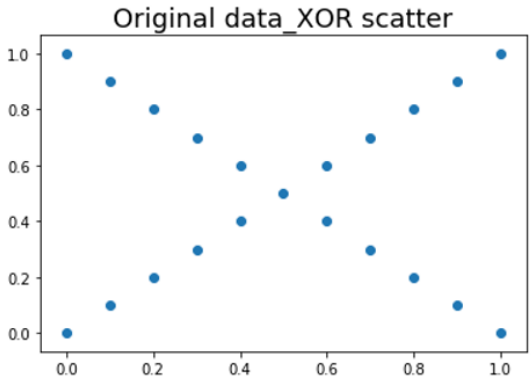


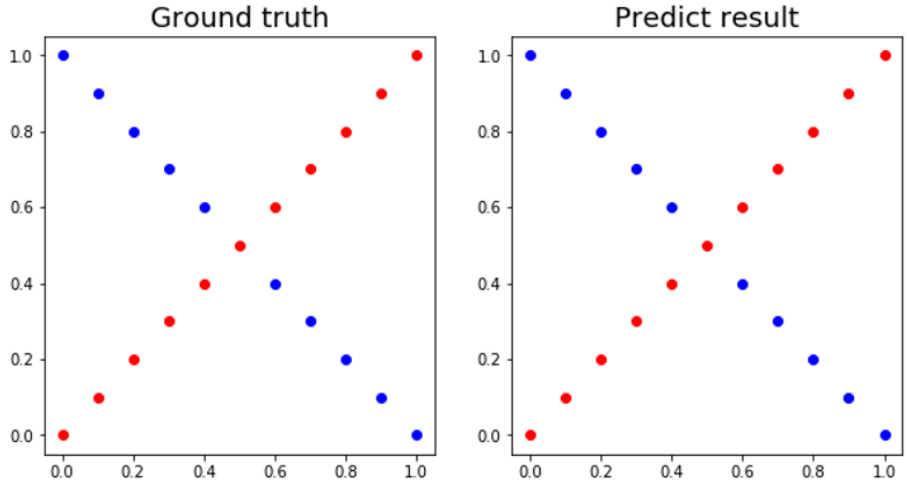


- For data type two：**XOR datasets**

```
epoch 10000 Loss: 0.00010023974959776331
epoch 15000 Loss: 6.351244294467604e-05
epoch 20000 Loss: 4.623399176118488e-05
epoch 25000 Loss: 3.624023744702336e-05
epoch 30000 Loss: 2.9744407812458264e-05
epoch 35000 Loss: 2.519157451459388e-05
epoch 40000 Loss: 2.1827469379777687e-05
epoch 45000 Loss: 1.9242584049847364e-05
epoch 50000 Loss: 1.7195667568856203e-05
epoch 55000 Loss: 1.5535487152962514e-05
epoch 60000 Loss: 1.4162489226198131e-05
epoch 65000 Loss: 1.30084969859355e-05
epoch 70000 Loss: 1.2025271833984456e-05
epoch 75000 Loss: 1.1177733377094332e-05
epoch 80000 Loss: 1.0439766869901044e-05
epoch 85000 Loss: 9.791535869887348e-06
epoch 90000 Loss: 9.217705729841147e-06
epoch 95000 Loss: 8.706237306850793e-06
epoch 100000 Loss: 8.247548374402405e-06
```

```
**-----Preduction Result-----**
[[0.0010625 ]
 [0.99846005]
 [0.00172944]
 [0.99845367]
 [0.00262537]
 [0.99834703]
 [0.00350959]
 [0.99803413]
 [0.00403401]
 [0.99401899]
 [0.00400774]
 [0.00352625]
 [0.99364381]
 [0.00284402]
 [0.99942972]
 [0.00218038]
 [0.9997104 ]
 [0.00163965]
 [0.99977253]
 [0.0012371 ]
 [0.99979084]]
**-------------------------------**
```





B. Show the accuracy of prediction

For data type one：**Linear datasets**	For data type two：**XOR datasets**

```
#predict y v.s label
def classify_correct(self, pred_y, label_y):
    correct_num = 0
    fail_pos = []
    print("**-----start to classify...-----**" )
    for i in tqdm(range(len(pred_y))):
        if(pred_y[i] > 0.5):
            pred_y[i] = 1
        else:
            pred_y[i] = 0

        if(pred_y[i] == label_y[i]):
            correct_num = correct_num + 1
        else:
            correct_num = correct_num
            fail_pos.append(i)
        sleep(0.01)
    print("**-----Classification Result-----**" )
    print("PASS : {} || FAIL : {} ".format(correct_num, len(pred_y)-correct_num) )
    if (len(pred_y)-correct_num)==0:
        print("FAIL position : NONE")
    else:
        print("FAIL position : {}".format(fail_pos))
    print("Accuracy : ", correct_num/len(pred_y))
    print("**--------------------------------**" )
```

```
**-----Classification Result-----**
PASS : 100 || FAIL : 0
FAIL position : NONE
Accuracy :  1.0
**--------------------------------**
```

```
**-----Classification Result-----**
PASS : 21 || FAIL : 0
FAIL position : NONE
Accuracy :  1.0
**--------------------------------**
```

C. Learning curve *(loss, epoch curve)*

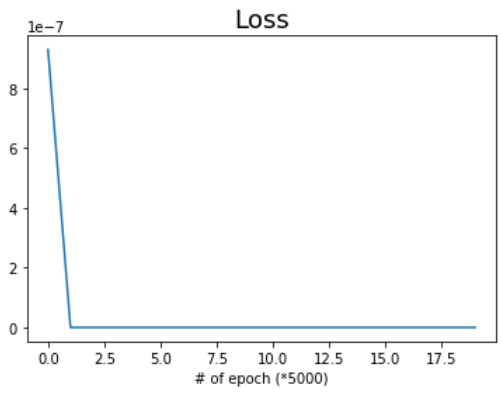
For data type one：**Linear datasets**	For data type two：**XOR datasets**

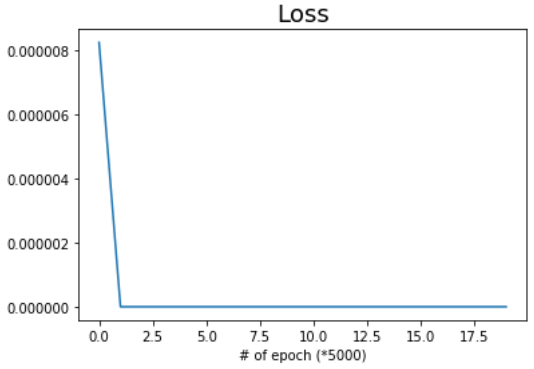
```
def plot_loss(self, loss, x , y):
    plt.title('Loss' , fontsize = 18)
    plt.plot(loss)

    pred_y = self.forward(x)
    print("**-----Preduction Result-----**")
    if(len(pred_y)==100):
        print(np.reshape(pred_y, (len(pred_y)//5, 5), order='F'))
    else:
        print(pred_y)
    print("**--------------------------------**" )
    self.classify_correct(pred_y, y)

    plt.xlabel("# of epoch (*" + str(num_to_show) + ")")
    plt.show()

    return pred_y
```





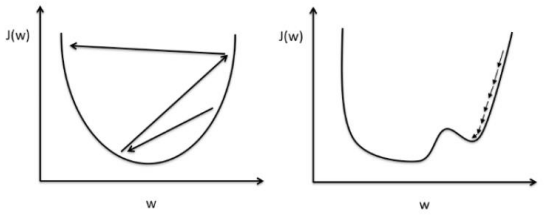
4. Discussion

A. Try different learning rate

(以 data type two：XOR datasets 為例)

Learning rates = 100	**Learning rates** = 10	**Learning rates** = 0.001
start training epoch 10000 Loss: 0.5238095238095238 epoch 15000 Loss: 0.5238095238095238 epoch 20000 Loss: 0.5238095238095238 epoch 25000 Loss: 0.5238095238095238 epoch 30000 Loss: 0.5238095238095238 epoch 35000 Loss: 0.5238095238095238 epoch 40000 Loss: 0.5238095238095238 epoch 45000 Loss: 0.5238095238095238 epoch 50000 Loss: 0.5238095238095238 epoch 55000 Loss: 0.5238095238095238 epoch 60000 Loss: 0.5238095238095238 epoch 65000 Loss: 0.5238095238095238 epoch 70000 Loss: 0.5238095238095238 epoch 75000 Loss: 0.5238095238095238 epoch 80000 Loss: 0.5238095238095238 epoch 85000 Loss: 0.5238095238095238 epoch 90000 Loss: 0.5238095238095238 epoch 95000 Loss: 0.5238095238095238 epoch 100000 Loss: 0.5238095238095238 Excution time : 7.688275 sec	start training epoch 10000 Loss: 0.5238095238095127 epoch 15000 Loss: 0.5238095238095127 epoch 20000 Loss: 0.5238095238095127 epoch 25000 Loss: 0.5238095238095127 epoch 30000 Loss: 0.5238095238095127 epoch 35000 Loss: 0.5238095238095127 epoch 40000 Loss: 0.5238095238095127 epoch 45000 Loss: 0.5238095238095127 epoch 50000 Loss: 0.5238095238095127 epoch 55000 Loss: 0.5238095238095127 epoch 60000 Loss: 0.5238095238095127 epoch 65000 Loss: 0.5238095238095127 epoch 70000 Loss: 0.5238095238095127 epoch 75000 Loss: 0.5238095238095127 epoch 80000 Loss: 0.5238095238095127 epoch 85000 Loss: 0.5238095238095127 epoch 90000 Loss: 0.5238095238095127 epoch 95000 Loss: 0.5238095238095127 epoch 100000 Loss: 0.5238095238095127 Excution time : 8.007854 sec	start training epoch 10000 Loss: 0.22229534659329925 epoch 15000 Loss: 0.19667173596489973 epoch 20000 Loss: 0.16152915661929287 epoch 25000 Loss: 0.1257537881377532 epoch 30000 Loss: 0.10103429076031449 epoch 35000 Loss: 0.08410671026625714 epoch 40000 Loss: 0.0718515673310085 epoch 45000 Loss: 0.06255432926850044 epoch 50000 Loss: 0.055214931922140886 epoch 55000 Loss: 0.04920662107335479 epoch 60000 Loss: 0.04410840760643758 epoch 65000 Loss: 0.039634028655887096 epoch 70000 Loss: 0.035597980832132746 epoch 75000 Loss: 0.031891645715476154 epoch 80000 Loss: 0.02846220256819143 epoch 85000 Loss: 0.025293323194895724 epoch 90000 Loss: 0.022388087431977483 epoch 95000 Loss: 0.01975517469573477 epoch 100000 Loss: 0.01739945065622362 Excution time : 8.051087 sec
-----Classification Result----- PASS : 10 || FAIL : 11 FAIL position : [0, 2, 4, 6, 8, 10, 11, 13, 15, 17, 19] Accuracy : 0.47619047619047616	**-----Classification Result-----** PASS : 10 || FAIL : 11 FAIL position : [0, 2, 4, 6, 8, 10, 11, 13, 15, 17, 19] Accuracy : 0.47619047619047616 **----------**	**-----Classification Result-----** PASS : 21 || FAIL : 0 FAIL position : NONE Accuracy : 1.0 **----------

learning rate 可以想像成在山谷中要找到最低點，每一次所跨出的步長，若太大的話容易像左下圖，發生震盪不容易找到最低點，而若設定太小則易發生右下圖得情況，除了訓練時間可能會很慢以外也可能發生在一定 epoch 內走不出局部低點的問題。



Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

由上方表格可以發現，當 learning rate 越小時(0.001)，收斂速度會較慢*(data 2 僅有 21 組數值，較不明顯)*，training 時較沒效率，且若設定過小如 0.0000001，會在我們指定的 epoch 發生無法收斂的問題導致錯誤!然而若設得太大(100.0)，則因為每次 W 更新值太大，反而會找不到 loss minimum 的位置，一直無法收斂。故設定適當的 learning rate 影響甚大，或是可以使用其他變動的 learning rate 去找尋最好的數值，有效提升 model training 效率和表現。

B. Try different numbers of hidden units

(以 data type two：XOR datasets 為例)

Hidden units = **5**00	Hidden units = **20**	Hidden units = **1**
start training epoch 10000 Loss: 0.23814085912398816 epoch 15000 Loss: 0.23812740487688192 epoch 20000 Loss: 0.23812026738543549 epoch 25000 Loss: 0.23811579381906375 epoch 30000 Loss: 0.23811271335148312 epoch 35000 Loss: 0.2381104617085584 epoch 40000 Loss: 0.23810874696250903 epoch 45000 Loss: 0.23810740075808254 epoch 50000 Loss: 0.23810631801788165 epoch 55000 Loss: 0.23810542933958004 epoch 60000 Loss: 0.23810468700820403 epoch 65000 Loss: 0.23810405725291167 epoch 70000 Loss: 0.2381035156346661 epoch 75000 Loss: 0.23810304414048758 epoch 80000 Loss: 0.2381026292776718 epoch 85000 Loss: 0.23810226078789606 epoch 90000 Loss: 0.23810193076149513 epoch 95000 Loss: 0.23810163301747547 epoch 100000 Loss: 0.23810136266358806 Excution time : 413.772991 sec	start training epoch 10000 Loss: 0.012584445916039164 epoch 15000 Loss: 0.004848684282595373 epoch 20000 Loss: 0.0026006987350890437 epoch 25000 Loss: 0.0016808279761453358 epoch 30000 Loss: 0.0012089894753325296 epoch 35000 Loss: 0.000930046399784224 epoch 40000 Loss: 0.0007487546731802588 epoch 45000 Loss: 0.0006227688254981072 epoch 50000 Loss: 0.0005307678966885006 epoch 55000 Loss: 0.0004609790895600618 epoch 60000 Loss: 0.0004064241696274679 epoch 65000 Loss: 0.0003627282473033708 epoch 70000 Loss: 0.00032702112514954143 epoch 75000 Loss: 0.00029734782747260075 epoch 80000 Loss: 0.00027233419530383885 epoch 85000 Loss: 0.00025098811932832755 epoch 90000 Loss: 0.00023257662096952425 epoch 95000 Loss: 0.0002165472058886178 epoch 100000 Loss: 0.00020247603357239933 Excution time : 7.988529 sec	start training epoch 10000 Loss: 0.2494557150215167 epoch 15000 Loss: 0.2494525531203861 epoch 20000 Loss: 0.24944979050871516 epoch 25000 Loss: 0.24944737968977493 epoch 30000 Loss: 0.2494452755125287 epoch 35000 Loss: 0.24944343626646528 epoch 40000 Loss: 0.24944182423145628 epoch 45000 Loss: 0.24944040583152066 epoch 50000 Loss: 0.24943915152245172 epoch 55000 Loss: 0.24943803551645216 epoch 60000 Loss: 0.24943703541983922 epoch 65000 Loss: 0.24943613183635108 epoch 70000 Loss: 0.2494353079699784 epoch 75000 Loss: 0.24943454924747405 epoch 80000 Loss: 0.2494338429710602 epoch 85000 Loss: 0.2494331780054384 epoch 90000 Loss: 0.24943254449915378 epoch 95000 Loss: 0.2494319336379662 epoch 100000 Loss: 0.24943133742657378 Excution time : 3.879416 sec
-----Classification Result----- PASS : 16 || FAIL : 5 FAIL position : [1, 3, 5, 7, 9] Accuracy : 0.7619047619047619 **-------------------------------**	**-----Classification Result-----** PASS : 21 || FAIL : 0 FAIL position : NONE Accuracy : 1.0 **-------------------------------**	**-----Classification Result-----** PASS : 11 || FAIL : 10 FAIL position : [1, 3, 5, 7, 9, 12, 14, 16, 18, 20] Accuracy : 0.5238095238095238 **-------------------------------**

由上述表格可發現，當 hidden units 太少時會導致 model 收斂不了，縱使運算時間超快，但結果其低，換言之 model 架構過於簡單，無法有效處理目前的 data，而當 hidden units 增加到足夠數目時就可以成功預測(如 20 個)，但是若加上過多*(重點在於過多)*hidden units 則會導致計算時間大幅增加，準確率也未必提高，甚至可能出錯。

C. Try without activation functions

Without activation function	Hidden units = **20**
epoch 0 Loss: 1.6918311398259622e+18 /usr/local/lib/python3.6/dist-package epoch 10000 Loss: nan epoch 15000 Loss: nan epoch 20000 Loss: nan epoch 25000 Loss: nan epoch 30000 Loss: nan epoch 35000 Loss: nan epoch 40000 Loss: nan epoch 45000 Loss: nan epoch 50000 Loss: nan epoch 55000 Loss: nan epoch 60000 Loss: nan epoch 65000 Loss: nan epoch 70000 Loss: nan epoch 75000 Loss: nan epoch 80000 Loss: nan epoch 85000 Loss: nan epoch 90000 Loss: nan epoch 95000 Loss: nan epoch 100000 Loss: nan Excution time : 3.353512 sec	start training epoch 0 Loss: 807.0965701699397 epoch 10000 Loss: 771.1266129789688 epoch 15000 Loss: 754.8161759347219 epoch 20000 Loss: 739.4104449808763 epoch 25000 Loss: 724.7883644652427 epoch 30000 Loss: 710.8509482942698 epoch 35000 Loss: 697.5165730693136 epoch 40000 Loss: 684.7174276862689 epoch 45000 Loss: 672.3967968549001 epoch 50000 Loss: 660.5069559472137 epoch 55000 Loss: 649.0075207658482 epoch 60000 Loss: 637.864140505057 epoch 65000 Loss: 627.0474528813511 epoch 70000 Loss: 616.5322418552552 epoch 75000 Loss: 606.2967535672524 epoch 80000 Loss: 596.3221370364865 epoch 85000 Loss: 586.5919841242384 epoch 90000 Loss: 577.091949123377 epoch 95000 Loss: 567.809432699817 epoch 100000 Loss: 558.7333181983595 Excution time : 4.068676 sec

類神經網絡中如果把 sigmoid function 去掉，意即不使用任何 activation functions，則在類神經網絡中便失去了非線性的效果，輸出的結果就都是之前 layer 輸入的線性組合作為這一層的輸出，換言之就是一班矩陣相乘的運算，輸出和輸入依然保持線性相關，做深度類神經網路便失去意義，也會導致有些分類問題會無法達成。由上方表格可以發現，如果把 sigmoid function 拿掉並搭配原本的 learning rate，則會使得 W 更新時變動數值過大(因為沒 sigmoid 把 model out 限制在 0~1)，導致整個 model 參數直接 overflow，基本上 train 不起來。若把 learning rate 調小(右側：learning rate 調成 0.0000001)，此時便能 train 出 model，然而因為沒有 sigmoid 來導入非線性，在解 XOR 問題時沒辦法只用線性 model 來分割 data,因此導致最後 performance 很差。

D. Gradient Problem

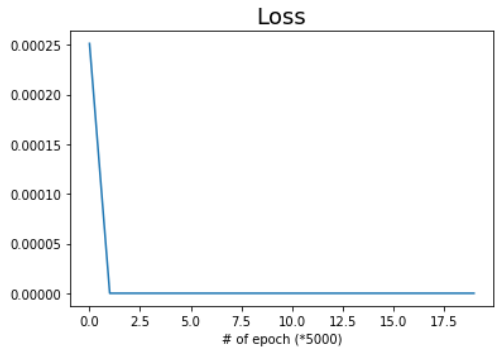
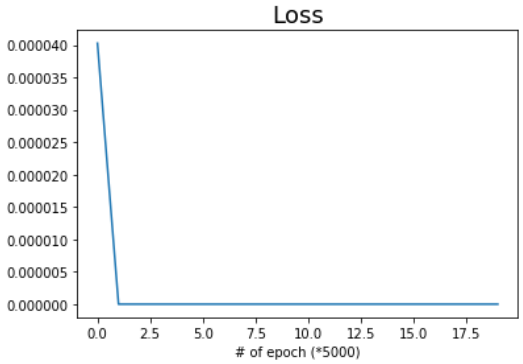
類神經網絡中 gradient 的問題是重要且常發生的，而此次 Lab 中 sigmoid function 就很容易出現梯度消失問題(Gradient Vanishing)，在神經網絡反向傳播中，梯度從後往前傳時，梯度不斷減小，最後變為零，此時，**淺層的神經網絡權重得不到更新**，那麼前面隱藏層的學習速率低於後面隱藏層的學習速率，即隨著**隱藏層數目的增加，分類準確率卻下降**，此現象叫做梯度消失。

梯度消失導致後層的權重更新的快，靠近輸出層的權值更新相對正常，而前層網絡由於梯度傳遞不過去而無法更新，靠近輸入層的權值更新會變得很慢，導致靠近輸入層的隱藏層權值幾乎不變，接近於初始化的權值，在網絡很深時，學習的速度很慢甚至無法學習，可知網絡深度越深問題會越來越嚴重，可用其他 activation function 替代、減少網絡深度、動態地改變 learning rate 等來解決。

5. Extra

B. Implement different activation functions

- **Activation func.：tanh**
- **learning rate：0.01**

For data type one：**Linear datasets**	For data type two：**XOR datasets**
start training epoch 0 Loss: 0.5099515079446219 epoch 10000 Loss: 0.0017303947157168273 epoch 15000 Loss: 0.0007911781853763988 epoch 20000 Loss: 0.0005790796865156063 epoch 25000 Loss: 0.0004928122368078818 epoch 30000 Loss: 0.00045229024456478346 epoch 35000 Loss: 0.0004369740441467817 epoch 40000 Loss: 0.0004254749669453841 epoch 45000 Loss: 0.00041132145623477915 epoch 50000 Loss: 0.0003962480692125413 epoch 55000 Loss: 0.0003807920032776111 epoch 60000 Loss: 0.00036515449942445266 epoch 65000 Loss: 0.0003495192369532441 epoch 70000 Loss: 0.0003340601831408744 epoch 75000 Loss: 0.00031892734992629464 epoch 80000 Loss: 0.0003042407283389271 epoch 85000 Loss: 0.00029009032814003875 epoch 90000 Loss: 0.00027653914017013116 epoch 95000 Loss: 0.0002636271618650297 epoch 100000 Loss: 0.0002513755387163278 Excution time：43.665555 sec	start training epoch 0 Loss: 0.7037057356225483 epoch 10000 Loss: 0.4761873188437281 epoch 15000 Loss: 0.4761867888597732 epoch 20000 Loss: 0.4761863241868597 epoch 25000 Loss: 0.4761858331560205 epoch 30000 Loss: 0.47618526791371785 epoch 35000 Loss: 0.476184582985836 epoch 40000 Loss: 0.4761837180304009 epoch 45000 Loss: 0.476182578564893 epoch 50000 Loss: 0.4761809993197171 epoch 55000 Loss: 0.4761786565592439 epoch 60000 Loss: 0.4761748132586268 epoch 65000 Loss: 0.47616734557810214 epoch 70000 Loss: 0.4761465752217566 epoch 75000 Loss: 0.4757718493911825 epoch 80000 Loss: 0.00030792909457377364 epoch 85000 Loss: 0.00012115792614505467 epoch 90000 Loss: 7.39087408938499e-05 epoch 95000 Loss: 5.245410563529994e-05 epoch 100000 Loss: 4.0270601219259415e-05 Excution time：14.505006 sec
-----start to classify...----- 100% 100/100 **-----Classification Result-----** PASS : 100 || FAIL : 0 FAIL position : NONE Accuracy : 1.0 **--------------------------------**	**-----start to classify...-----** 100% 21/21 **-----Classification Result-----** PASS : 21 || FAIL : 0 FAIL position : NONE Accuracy : 1.0 **--------------------------------**
	

- **Activation func.：ReLu**
- **learning rate：0.0015**

For data type one：**Linear datasets**	For data type two：**XOR datasets**
start training epoch 0 Loss: 0.51 epoch 10000 Loss: 0.51 epoch 15000 Loss: 0.51 epoch 20000 Loss: 0.51 epoch 25000 Loss: 0.51 epoch 30000 Loss: 0.51 epoch 35000 Loss: 0.51 epoch 40000 Loss: 0.51 epoch 45000 Loss: 0.51 epoch 50000 Loss: 0.51 epoch 55000 Loss: 0.51 epoch 60000 Loss: 0.51 epoch 65000 Loss: 0.51 epoch 70000 Loss: 0.51 epoch 75000 Loss: 0.51 epoch 80000 Loss: 0.51 epoch 85000 Loss: 0.51 epoch 90000 Loss: 0.51 epoch 95000 Loss: 0.51 epoch 100000 Loss: 0.51 Excution time：10.162688 sec	start training epoch 0 Loss: 0.47619047619047616 epoch 10000 Loss: 0.47619047619047616 epoch 15000 Loss: 0.47619047619047616 epoch 20000 Loss: 0.47619047619047616 epoch 25000 Loss: 0.47619047619047616 epoch 30000 Loss: 0.47619047619047616 epoch 35000 Loss: 0.47619047619047616 epoch 40000 Loss: 0.47619047619047616 epoch 45000 Loss: 0.47619047619047616 epoch 50000 Loss: 0.47619047619047616 epoch 55000 Loss: 0.47619047619047616 epoch 60000 Loss: 0.47619047619047616 epoch 65000 Loss: 0.47619047619047616 epoch 70000 Loss: 0.47619047619047616 epoch 75000 Loss: 0.47619047619047616 epoch 80000 Loss: 0.47619047619047616 epoch 85000 Loss: 0.47619047619047616 epoch 90000 Loss: 0.47619047619047616 epoch 95000 Loss: 0.47619047619047616 epoch 100000 Loss: 0.47619047619047616 Excution time：6.046147 sec
-----start to classify...------ 100% 100/100 **-----Classification Result-----** PASS : 49 || FAIL : 51 FAIL position : [0, 1, 8, 10, 11, 13, 14, 16, 9, 60, 61, 62, 63, 66, 67, 68, 69, 70, 71, 73, Accuracy : 0.49 **-------------------------------**	**-----start to classify...------** 100% 21/21 **-----Classification Result-----** PASS : 11 || FAIL : 10 FAIL position : [1, 3, 5, 7, 9, 12, 14, 16, Accuracy : 0.5238095238095238 **-------------------------------**

- **Activation func.：leaky-relu**
- **learning rate：0.01**

For data type one：**Linear datasets**	For data type two：**XOR datasets**
start training epoch 0 Loss: 0.46331915678107094 epoch 10000 Loss: 0.04851203720398631 epoch 15000 Loss: 0.048301828039145435 epoch 20000 Loss: 0.04828052755787345 epoch 25000 Loss: 0.04829789997382557 epoch 30000 Loss: 0.04828153794396225 epoch 35000 Loss: 0.04828419138758234 epoch 40000 Loss: 0.04828224660051089 epoch 45000 Loss: 0.048279542695493435 epoch 50000 Loss: 0.04827943025626477 epoch 55000 Loss: 0.048278432740626126 epoch 60000 Loss: 0.04828377978596306 epoch 65000 Loss: 0.048280467829079604 epoch 70000 Loss: 0.04828147554718223 epoch 75000 Loss: 0.04827727458109923 epoch 80000 Loss: 0.04827787676660151 epoch 85000 Loss: 0.0482768845347299 epoch 90000 Loss: 0.048279103923220726 epoch 95000 Loss: 0.04827784409890318 epoch 100000 Loss: 0.04827603842800422 Excution time：18.891298 sec	start training epoch 0 Loss: 1.0328838129324671 epoch 10000 Loss: 8.801357635404639e-12 epoch 15000 Loss: 1.2131139086830287e-15 epoch 20000 Loss: 1.6716657432460788e-19 epoch 25000 Loss: 2.3037017697468268e-23 epoch 30000 Loss: 3.352741645995429e-27 epoch 35000 Loss: 3.0641753461490387e-30 epoch 40000 Loss: 2.3277346589450184e-30 epoch 45000 Loss: 2.3277346589450184e-30 epoch 50000 Loss: 2.3277346589450184e-30 epoch 55000 Loss: 2.3277346589450184e-30 epoch 60000 Loss: 2.3277346589450184e-30 epoch 65000 Loss: 2.3277346589450184e-30 epoch 70000 Loss: 2.3277346589450184e-30 epoch 75000 Loss: 2.3277346589450184e-30 epoch 80000 Loss: 2.3277346589450184e-30 epoch 85000 Loss: 2.3277346589450184e-30 epoch 90000 Loss: 2.3277346589450184e-30 epoch 95000 Loss: 2.3277346589450184e-30 epoch 100000 Loss: 2.3277346589450184e-30 Excution time：12.831198 sec
-----start to classify...------ 100% 100/10 **-----Classification Result-----** PASS : 93 || FAIL : 7 FAIL position : [14, 33, 45, 54, 76, 86, 90] Accuracy : 0.93 **-------------------------------**	**-----start to classify...------** 100% 21/21 **-----Classification Result-----** PASS : 21 || FAIL : 0 FAIL position : NONE Accuracy : 1.0 **------------------------------**