DLP Lab1 Backpropagation

0711529 陳冠儒

1. Introduction (20%)

建造一個具有兩層hidden layers的model，並計算forward和backward propagation，來預測Linear和XOR dataset。

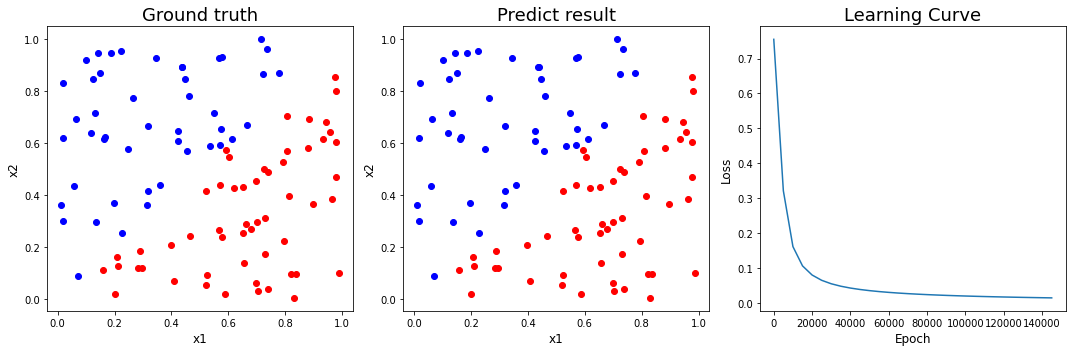
Dataset：

x: shape = (number of data, input dimension)

y: shape = (number of data, label)

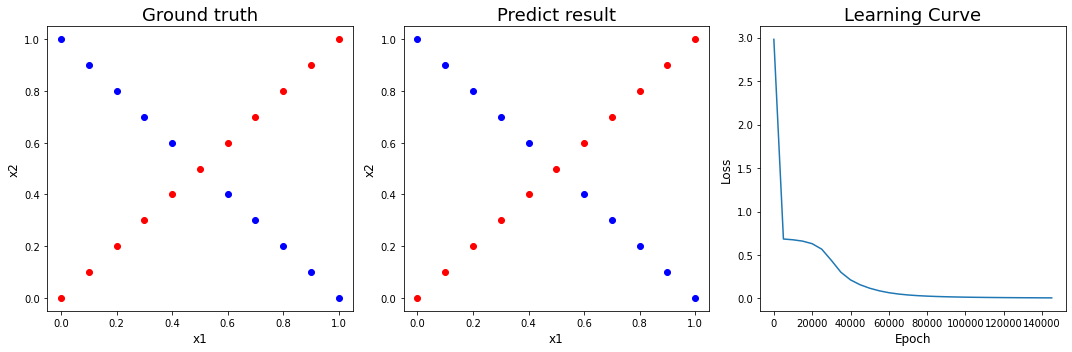
1. Linear

* label = 0, x1 > x2 which falls on the bottom-right side.
* label = 1, x1 < x2 which falls on the upper-left side.



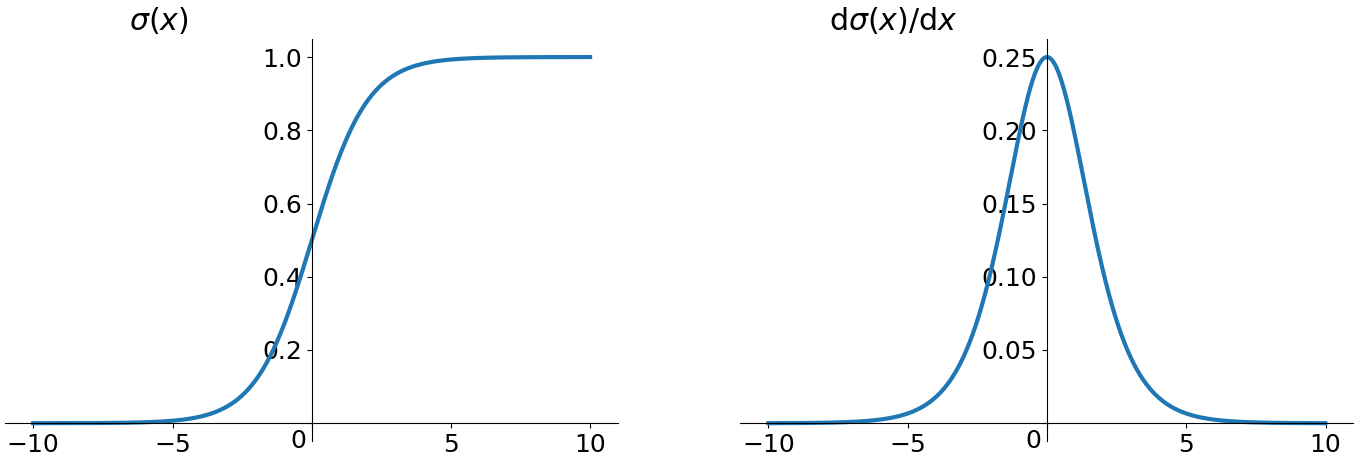
1. XOR

* label = 0, x2 = x1
* label = 1, x2 = 1 - x1



1. Experiment setups(30%):
   1. Sigmoid functions

* 簡介



Sigmoid functions可以用來當作hidden layer的activation function來使其成non-linear，也可以用在output layer使其成Bernoulli output distributions。

* Sigmoid function在數學上的介紹

Sigmoid function是logistic function的一個特例。

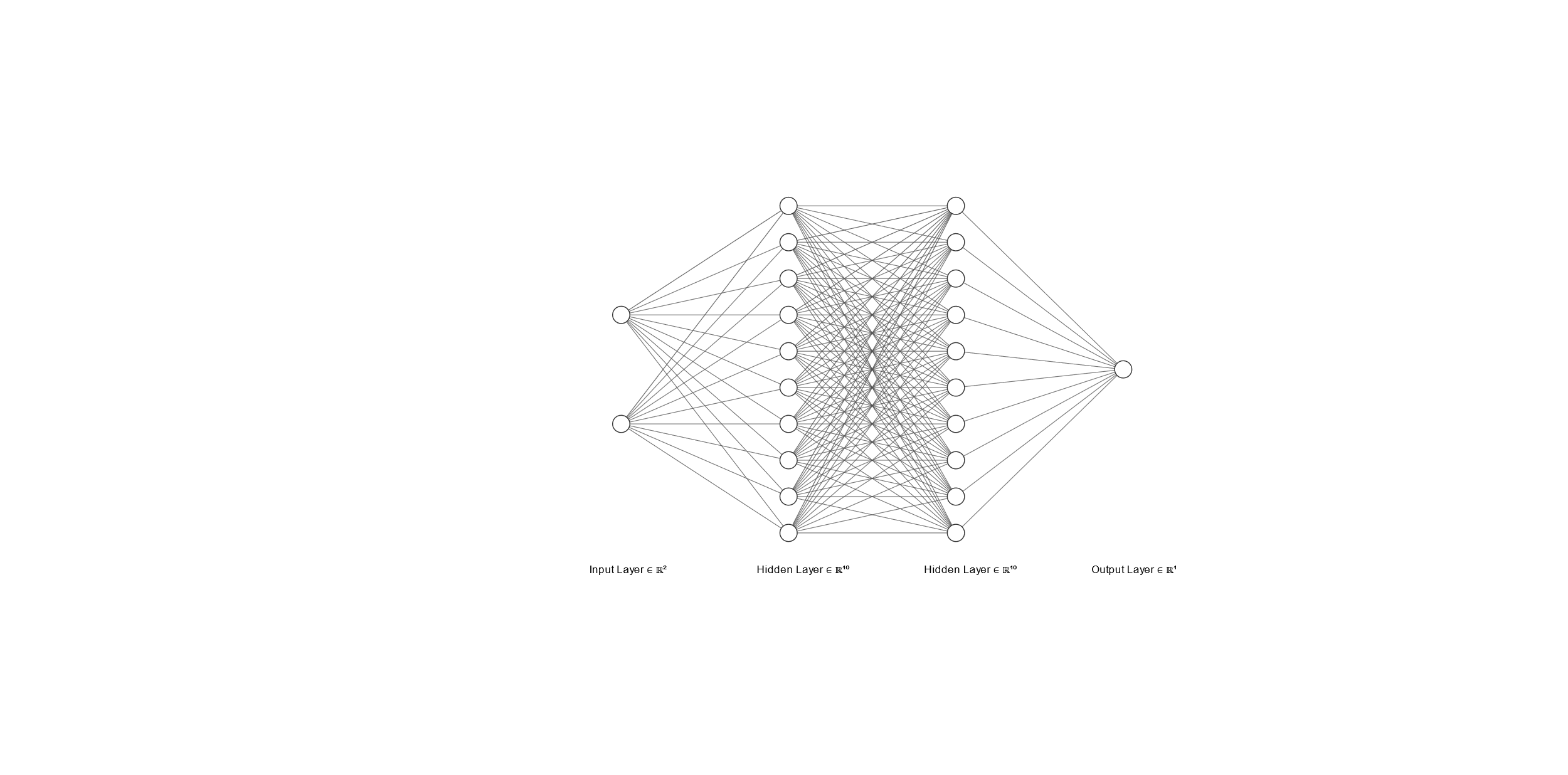
L控制最大最小值、k為最小到最大的變化速率、為x的偏移量，而sigmoid function就是當L = 1、k = 1且=0，故在output layer用sigmoid function的二分法model也被稱為logistic regression。

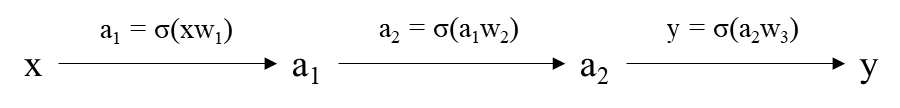
* Derivation of sigmoid function
* Sigmoid優點

中央區的信號增益較大，對兩側區的信號增益小，在信號的特徵空間映射上佳，學習時可以將重點特徵推向中央區，將非重點特徵推向兩側區。

* Sigmoid缺點

1. 均值不為0，意味著自帶了一個bias，在計算時是額外的負擔，這會使得收斂變得更慢。
2. 在其飽和區會變化的很緩慢，微分趨近於0，容易造成梯度消失，使得訊息丟失。
   1. Neural network





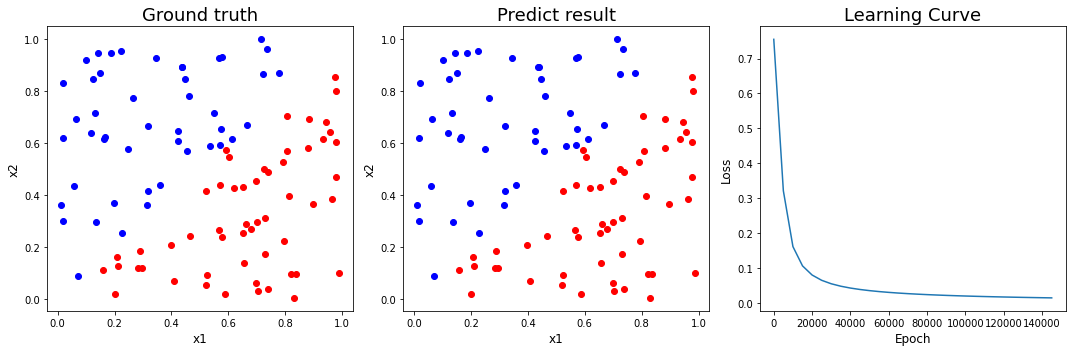
* Neural Architecture & parameters design
* hidden units ( 10, 10 )
* learning rate 0.01
* Epoch 150000
* Training data number: (1.) linear: 100, (2.) XOR: 21
* Testing data number: (1.) linear: 100, (2.) XOR: 61
* Loss Function

我使用Maximum Likelihood來當作loss function，上式所代表的是兩個Bernoulli distribution的cross entropy，即算出y distribution和ypred distribution的距離，minimizing cross entropy就是希望這兩個越接近越好。

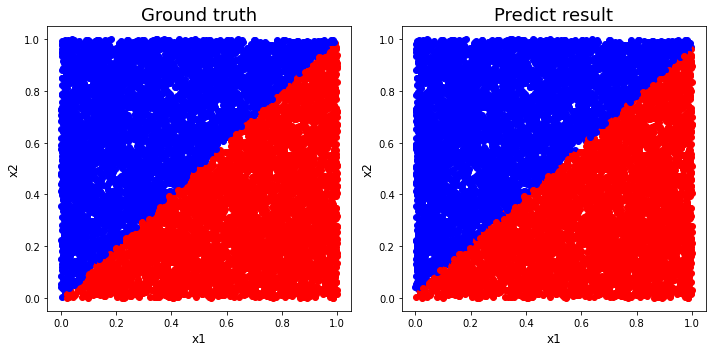
* 1. Backpropagation
* , where
* , where
* , where

1. Results of your testing (20%)
   1. Screenshot and comparison figure
      * Linear  
        Testing data generating by changing the random seed

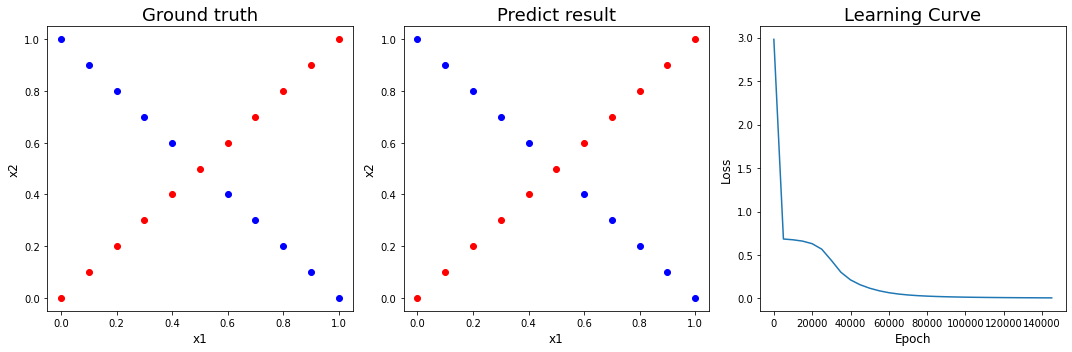
* n\_data = 100 → accuracy = 1.0



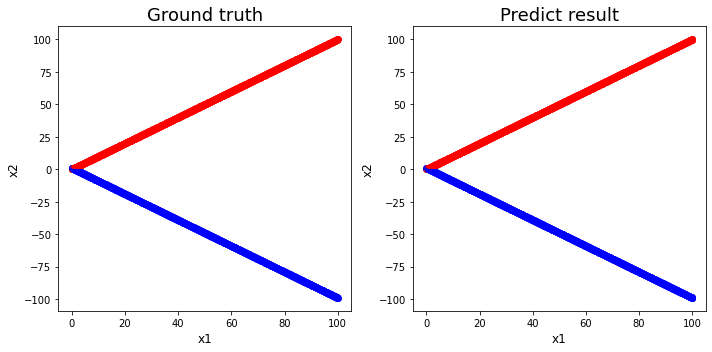
* n\_data = 10000 → accuracy = 0.984



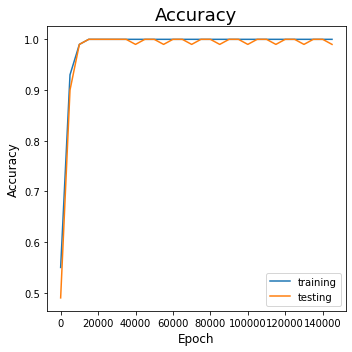
* + - XOR  
      Testing data generating by changing the number of points the function generated.
* n\_data = 21 → accuracy = 1.0



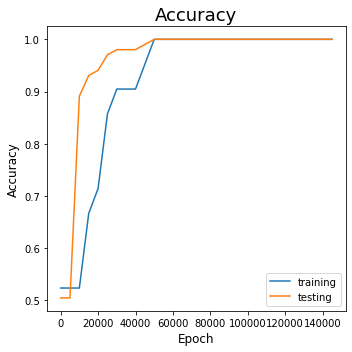
* n\_data = 2001 → accuracy = 1.0



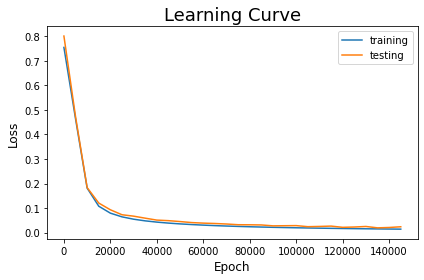
* 1. Show the accuracy of your prediction
     + Linear



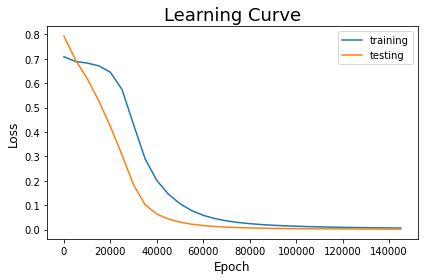
* + - XOR



* 1. Learning curve (loss, epoch curve)
     + Linear



* + - XOR



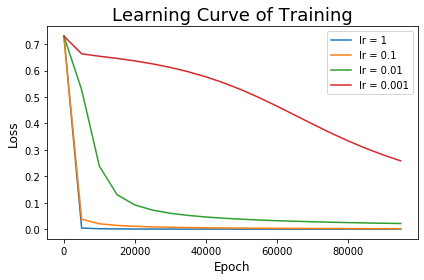
* 1. Anything you want to present
* Discuss about the learning curve and accuracy of XOR

XOR function accuracy跟learning curve testing都優於training的原因，是因為其function較為簡單學習，且dataset是依照同一個線性function所產生的，所以在training時，因為只有21個資料點，會由於樣本少，而使得每一個錯誤對cost和accuracy的影響被放大，但在testing中，因為資料點多(101個資料點)，所以錯誤的影響會被稀釋。

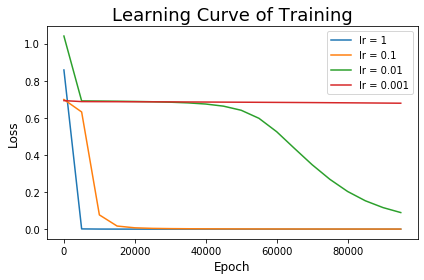
1. Discussion (30%)
   1. Try different learning rates

固定Epoch 100000、hidden units ( 5, 5 )。

1. Linear



1. XOR

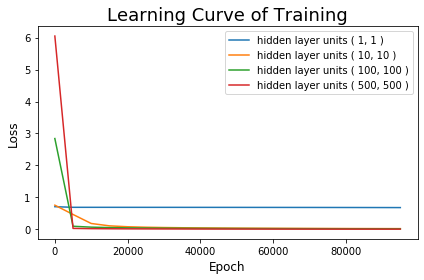


結論：從上面linear跟XOR不同learning rate造成的learning curve of training可以證實以下幾點。

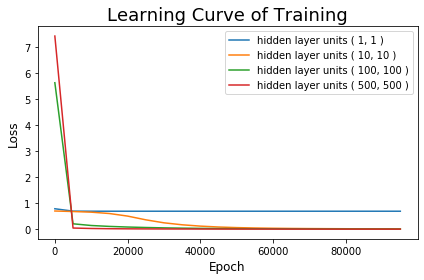
1. learning rate過大的話，他在一個epoch改變的很大，因此收斂的速度快，但是過大的learning rate可能因為步伐太大而無法收斂到最小點 (因為上面所求的function都較為簡單因此不會看到此現象)。
2. learning rate過小的話，他每一個epoch改變的都非常小，因此需要很久很久才會收斂，且收斂時如果進入到了local minimum就可能無法走出去了。
3. learning rate適中的話，收斂速度不慢，且可以進入到global minimum。
   1. Try different numbers of hidden units

固定Epoch 100000、Learning rate 0.01。

1. Linear

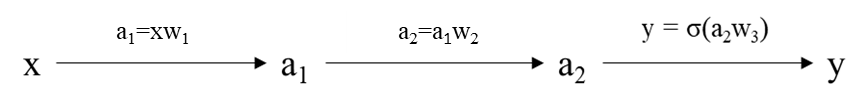


1. XOR



結論：從上面linear跟XOR不同hidden layer units造成的learning curve of training可以證實以下幾點。

1. hidden layer units越多neural network的功能會越強大，能夠擬合越多的function，故training時可以讓loss降得很低，但是也有可能會產生overfitting，因此多的hidden layer units可以搭配著regularization一起使用，雖然會train比較久但是可以得到比較好的結果。
2. hidden layer units太少的話，雖然因為運算可已很快，但是可能會造成underfitting，因此最好選擇適中的hidden layer units，以在運算時間和fitting結果中達到平衡。

* 1. Try without activation functions

Backpropagation

* + - , where
    - , where
    - , where

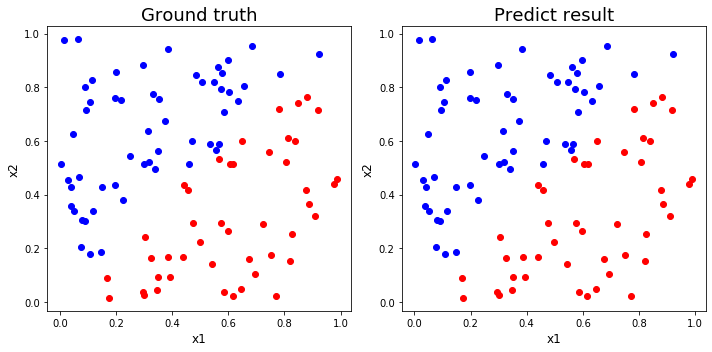
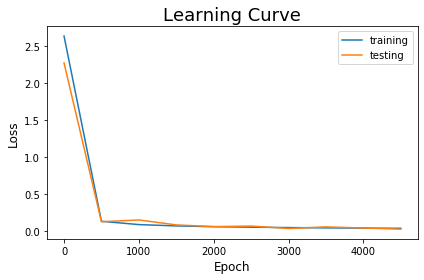
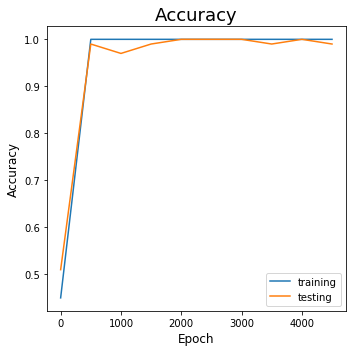
其他的參數使用同最上面的neural network parameters設定

* hidden units ( 10, 10 )
* learning rate 0.01
* Training data number: (1.) linear: 100, (2.) XOR: 21
* Testing data number: (1.) linear: 100, (2.) XOR: 61

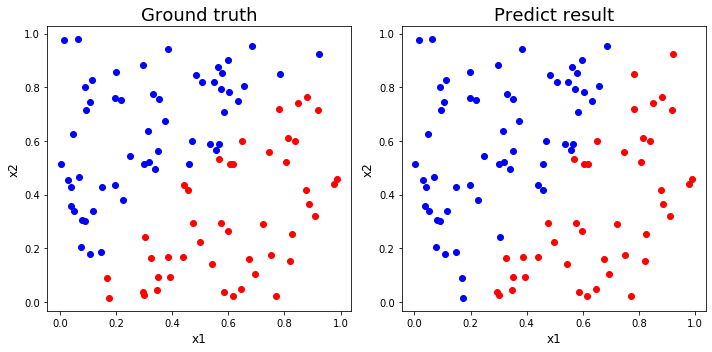
1. Linear

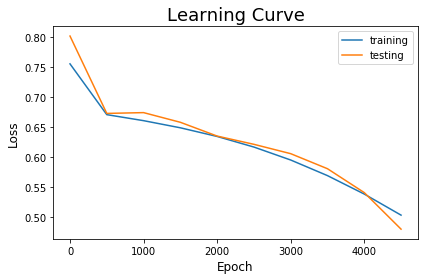
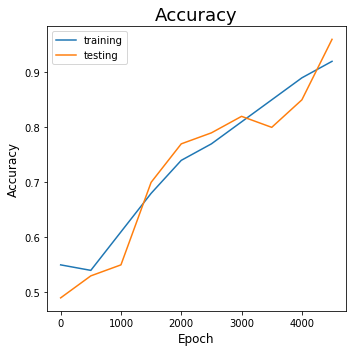
在小的epoch沒有activation function的network反而比有用sigmoid當activation function還有有效，因為這個dataset本身就是要訓練出一條linear的function，所以是不用添加sigmoid function使其變成non-linear的，而添加non-linear的activation function後，也是可以訓練出來的，只是會需要更多次的epoch去fit他。

* Epoch 5000 without activation function

* Epoch 5000 with activation function sigmoid

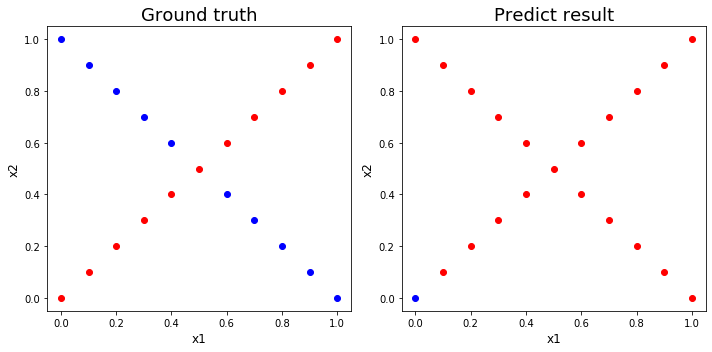


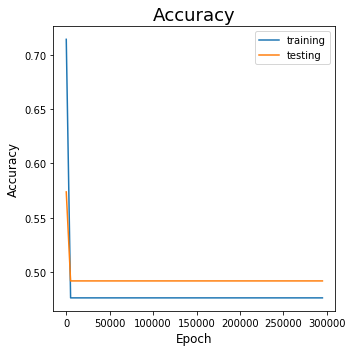
 

1. XOR

正如AI winter時，Minsky and Papert的《Perceptrons》書中所述，XOR因為不是線性的問題，所以不論用再多的epoch或是再小的learning rate去train，都不會得到正確的答案。

* Epoch 30000 without activation function

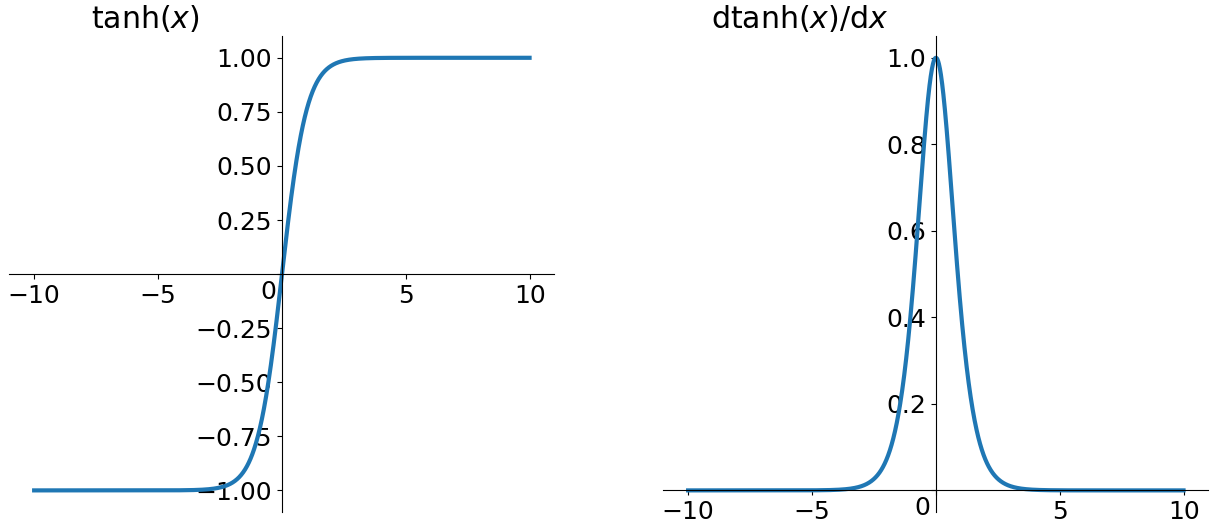


* 1. Anything you want to share
* Use Different Activation Function at hidden layer

1. tanh

* Derivation of sigmoid function



* tanh優點

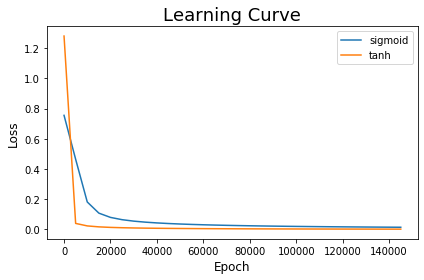
zero-centered，其均值是0，沒有自带bias，在計算時是額外負擔少，這會使得收斂變得更快。

* tanh缺點

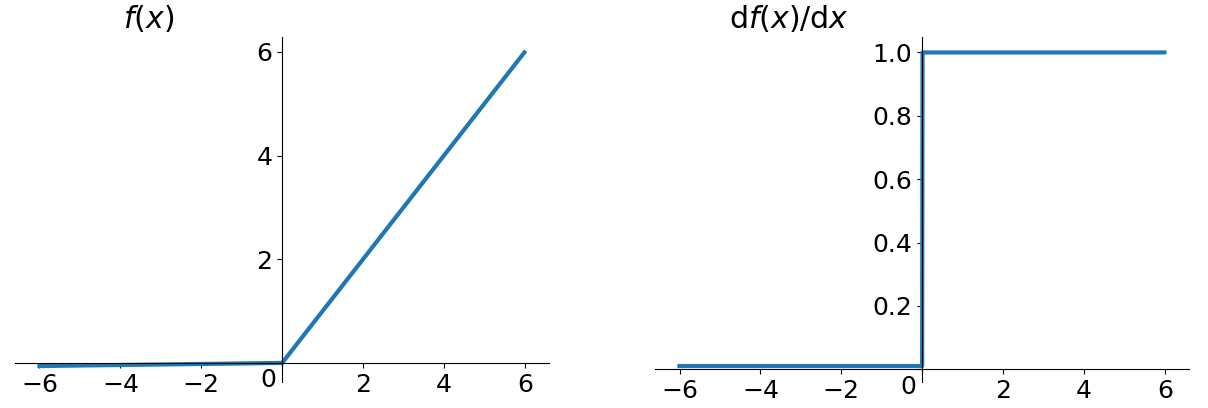
在其飽和區的接近於0，都容易產生後續梯度消失、計算量大的問題。

* 套用到model

我將其套用到linear neural network model的hidden layer中，其training loss與sigmoid比較(其他參數固定下)，可以發現確實收斂的比sigmoid更快。



1. Leaky ReLU



* Leaky ReLU優點

基本上繼承了ReLU的所有優點

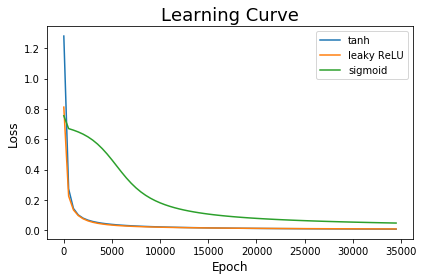
1. 解決了gradient vanishing問題(在正區間)
2. 計算速度非常快，收斂速度遠快於sigmoid和tanh
3. 解決Dead ReLU Problem

* Leaky ReLU缺點

在實際操作當中，並沒有完全證明Leaky ReLU總是好於ReLU。

* 套用到model

我將其套用到linear neural network model的hidden layer中，其training loss與sigmoid比較(其他參數固定下)，可以發現確實收斂的比sigmoid更快，但可能因為所求function較為簡單，因此跟tanh差不多。



1. Reference

* 也談激活函數Sigmoid, Tanh, ReLu, softplus, softmax：<https://zhuanlan.zhihu.com/p/48776056>
* 聊一聊深度學習的activation function： https://zhuanlan.zhihu.com/p/25110450