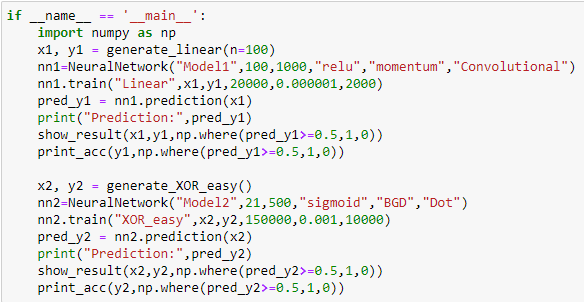
1. Introduction

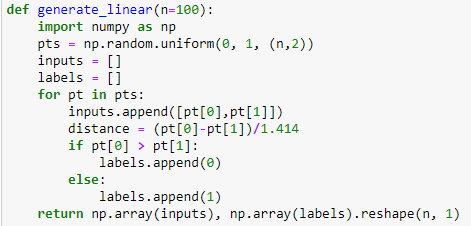
(a)這份作業的目的是透過手刻深度學習model，來對測資分別做一個分類器

(b)先看main

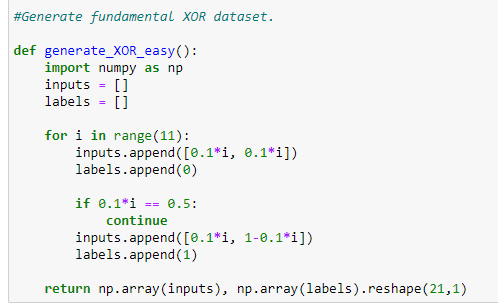


很清楚看到，本次作業主要能分成建立data，建模型，train和test的部份。

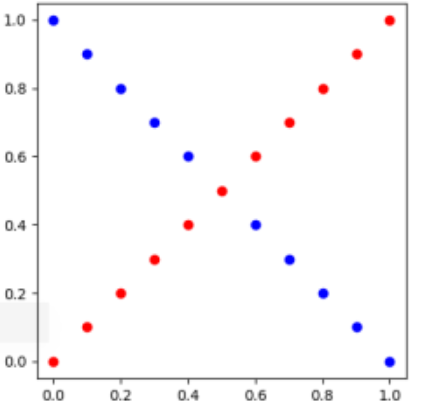
(c)創立data的程式碼



第一個測資是隨機生成二維點X，如果x1+x2>0.5算成藍的，反之紅的。



第二個測資是簡易版XOR 共21個點，紅藍分類如下



(d)Initialize Model

接下來是建模型的部份，我的model算滿自由的。

hidden\_layer1固定是dot layer(就高中學的矩陣乘法，我實際上不知道這層layer叫甚麼，因為numpy中是呼叫np.dot，姑且叫他dot layer)

hidden\_layer2可選擇要convolution還是dot layer

activation layer1 可選擇要relu還是sigmoid還是都不要

activation layer2 固定sigmoid

optimizer可選擇要BGD(類似SGD，但我的input都是一組測資，所以自然是BGD)或是momentum

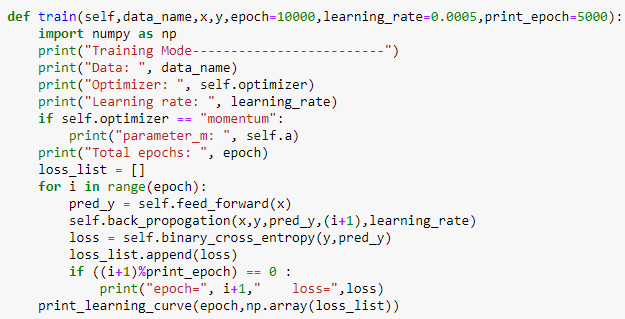
然後hidden layer和learning rate部分能自由調整

initialize的東西有點多

詳細可以看我的code。

(e)Training

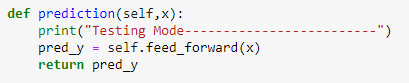
接下來就是train，主要能分forward，backpropagation和gradiant descent(我把他包在function self.back\_propagation裡)，還有順便畫learning curve，後面再提細節。



(f)Test

test又能分成predict和evaluation

predict就是將train好的model再做一次forward，然後機率>=0.5算藍，<0.5算紅



evaluation的部份，我有輸出acc和ground\_truth/predict的比較圖

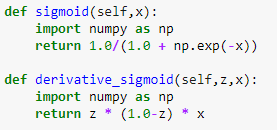




1. Experiment setups

(a)sigmoid functions

(1)sigmoid



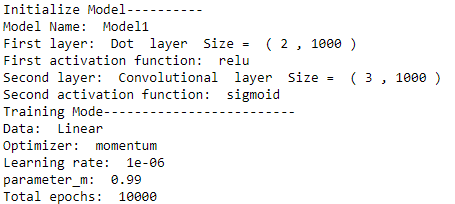
forward就1/1+exp(-x)

back propagation就1\*(1-z)，有連鎖率的話就多乘個連鎖率x

(b)Neural network

因為有兩組測資，然後我想順便寫加分題，所以我為兩組測資分別寫了不同的model和training方法。

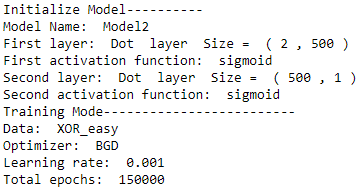
第一組測資是linear，我使用的網路規格和training規格如下:



loss評分標準是用binary cross entropy

input的話每次就是全部測資(100,2)

第二組測資是XOR，我使用的網路規格和training規格如下:

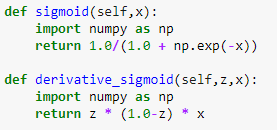


loss評分標準是用binary cross entropy

input的話每次就是全部測資(21,2)

(C)Back\_propagation

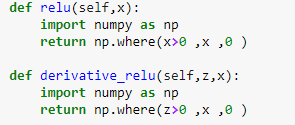
(1)sigmoid



forward就1/1+exp(-x)

back propagation就1\*(1-z)，有連鎖率的話就多乘個x

(2)relu



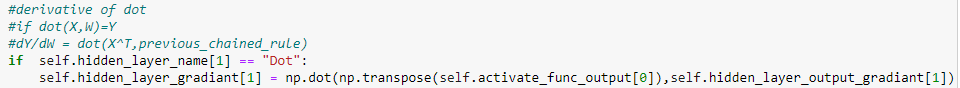
relu的forward就是max(x,0)

但back propagation在0時不能微分，但還好只有一個點，所以姑且令back propagation的結果是1 if x>0， 0 if x<=0。然後多一個x是因為連鎖率的關係

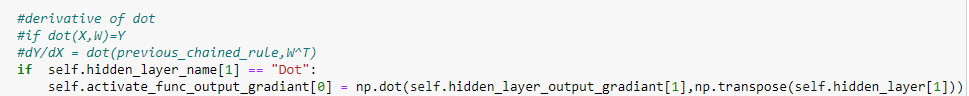
(3)Dot layer

假設輸入是X，dot layer是W，輸出就是XW=Y(或WX，但為了實作方便我是用XW)

Dot layer對W的back propagation是transpose(X)乘上前面連鎖率的東西



Dot layer對X的back propagation是前面連鎖率的東西乘上transpose(W)



(4)convolution layer

假設輸入是X，Convolution layer是W，輸出就是X conv W=Y(或W conv X，但為了實作方便我是用XW)

Convolution layer對W的back propagation是X conv 前面連鎖率的東西

Convolution layer對X的back propagation是前面連鎖率的東西 conv 轉180度的X矩陣，然後要另外padding，大小才會對

但Convolution layer對X的back propagation實作太麻煩，所以我就另外刻一個演算法一格一格算了，因為自己刻的，所以速度慢了不少。

reference:<https://blog.csdn.net/Libo_Learner/article/details/84556017>

code有點亂，就不放上來了。

詳細code請看我的source code。

(5)binary cross entropy

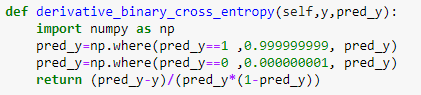
一個在binary classification情況下，好的loss計算方法。

公式如下:



back propagation公式如下:

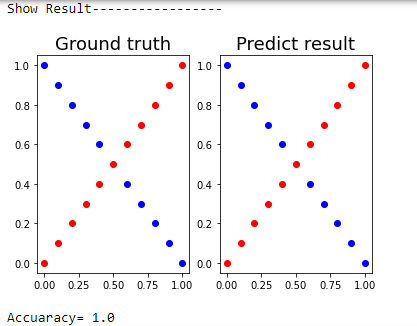
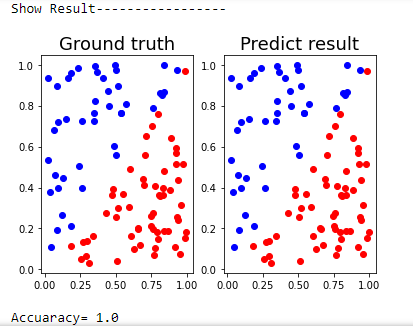




3.Results of your testing

(a) Screenshot and comparison figure

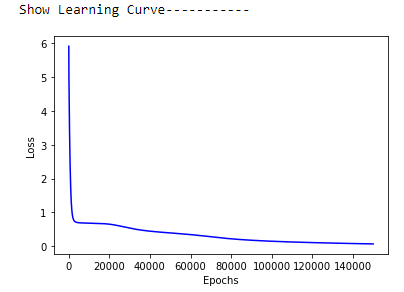
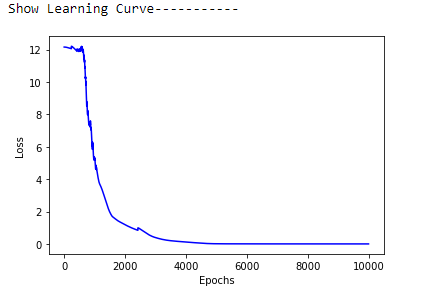
(b) Show the accuracy of your prediction



(a)(b)圖片連在一起，只好一起寫了

(c)Learning curve (loss, epoch curve)

linear:(optimizer=momentum) XOR:(optimizer=BGD)



(d)anything you want to present

從3.(c)的learning curve可以看出一些東西

就是optimizer用momentum在初期因為劇烈震盪，所以loss初期會在一個高值，而且learning rate很容易沒調好就很容易train爆(我目前train爆的機率大概1~5% 但為了加分題只能這樣了...)，而BGD比較沒這問題。但是momentum的好處顯現在後期，基本上training的時間夠久，表現通常都比XOR好。

4.Discussion

(a)Try different learning rates

這部分我想分享momentum的learning rates，因為momentum除了learning rates外，還有一個hyperparameter\_m，而learning rate取決於hyperparameter\_m的衰退速度。

為了展現momentum與BGD的不同，還有為了展現learning rates的重要性，我故意將hyperparameter\_m設到0.99，learning rate設到10^-6。

其實我有想要克服3(c)圖中 momentum在初期的劇烈震盪，所以我有直接將momentum調到10^-7，會發現收斂速度變很慢，而且容易收斂在local minimum，這完全與momentum的動機相悖。

當然我也試著將learning rate設大一點(10^-5)，結果就是gradiant太高，loss劇烈震盪卡在高值。

結論是learning rate太大太小都不行。

(b)Try different numbers of hidden units

這份作業一開始我也不知道hidden units要用多大才好，所以我就照著我的XOR的那個model寫(dot layer -sigmoid - dot layer -sigmoid)。然後dot layer分別用過2\*4,4\*1(因為教授講義例題這樣寫) 和2\*10000,10000\*1，最後發現都train不太起來，loss降不下來，還以為是我back propagation刻錯。

現在想了想滿合理的，參數太少的話，model含有的資訊量太少，當然訓練不起來。

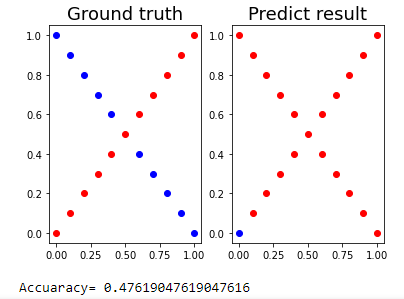
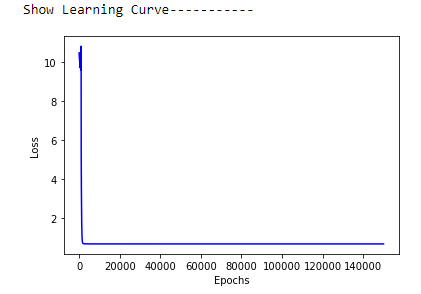
參數太多的話，我猜是因為內含太多重複，無用的資訊，導致網路無法有效整理資訊。

(c)Try without activation functions

因為輸出想弄在[0,1]間，所以能移除的activation function只能是兩層hidden layer中間的那個activation funciton。

於是我將XOR那組model的第一個activation function移除，結果就是收斂到一定程度後降不下去。

其實這還滿合理的，因為兩個線性矩陣相乘還是線性矩陣，還不如只疊一層。而有人實驗證明，至少兩層hidden layer+activation function才能應付大多簡單分類情形。



(d)Anything you want to share

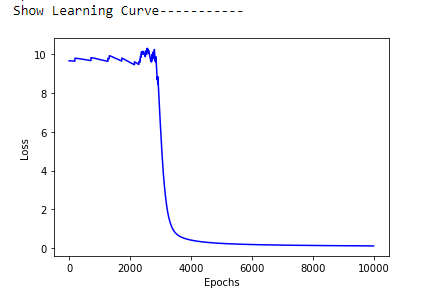
1. relu真的好用

就算真的沒在loss上看出和sigmoid的差異，他的運算量非常小，要不是convolution手刻算太慢，不然linear的model早就能train個100000epochs了。

(2)convolution layer搭配momentum的效果

從3(c)的linear的loss圖，已經看過convolution配momentum的效果。

接著看下面這張loss圖，把linear的model換成兩層dot layer。



可以很清楚發現，沒有convolution layer的loss curve容易初期更容易在高值徘徊，還有不少次在epoch=30000,40000才開始收斂。換言之，convolution可以有效減少momentum初期在高值徘徊的問題，我認為這是因為convolution同時收取了不同排data的資訊，更能幫助模型統整資料的特性。

但也有個小缺點，就是convolution在收斂時偶爾會跳起來，不過目前沒看到有甚麼不良影響。

5.

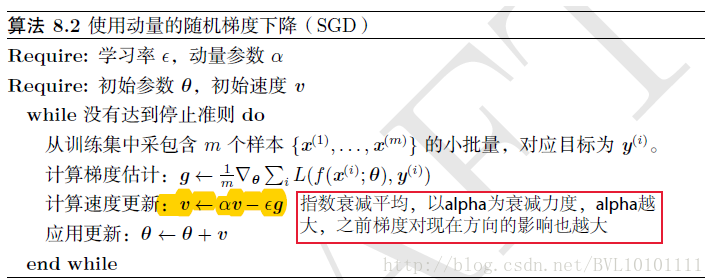
(a)Implement different optimizers.

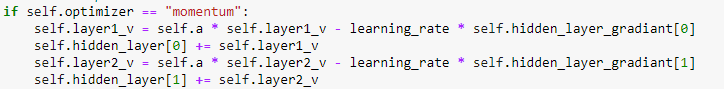
Momentum

BGD的壞處就是容易收斂到local minimum，而momentum靈感來自現實生活中的物理世界，可以翻過local minimum，去到更低的地方。

所以我在XOR的模型使用BGD外，我又在我Linear的模型另外寫了Momentum。

公式:





(b)Implement different activation functions.

我在Linear的模型加了一層relu，發現速度比sigmoid快上很多。

Forward，Back propagation和實作在2.(c)已經提過。

(c)Implement convolutional layers.

我在Linear的模型加了一層convolution，發現因為同時吸收多格的資訊，可以幫助momentum早期快速收斂。

Forward，Back propagation在2.(c)