人工智慧 作業一

資工112 40847015S 紀軒宇

1. 首先，請說明你所用之機器軟硬體規格、所用的作業系統等相關資訊以及你為何選擇這樣的規格。另外請提供你的聯絡電話，以便不時之需。

CPU: Intel® Core™ i7-8750H

GPU:Nvidia GeForce GTX 1060

ram:24g ddr4 2666

OS: Windows 10

會選擇這樣的規格是因為當時考量未來可能會學習機器學習相關知識，選擇了有獨立顯示卡的電腦，方便日後學習。

連絡電話：0988-056-412

1. 「實作類神經網路計算神秘函數」講義第6頁的程式，只訓練了200次。請改寫程式以訓練不同的次數並做出誤差圖表。請問誤差會無限的小下去嗎？如果改寫程式加入learning rate的話，會有差別嗎？（請提供你的程式）

程式部分

from numpy import \*

import matplotlib.pyplot as plt

import math

import time

train\_in = array([[0, 2, 0], [0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]])

train\_sol = array([[0, 0, 0, 1, 1]]).T

def epoch\_error(epoch, learning\_rate):

  random.seed(1)

  nn\_weights = 2 \* random.random((3, 1)) - 1

  # print(nn\_weights)

  x = range (epoch)

  y = []

  y\_test = []

  min\_train\_loss = 999

  start = time.time()

  # print(sum(abs(train\_sol-(1/(1+exp(-(dot(train\_in, nn\_weights)))))))/5)

  for i in range (epoch):

    # print("\n i=", i, "\nnn\_weight=\n", nn\_weights)

    train\_out = 1/(1+exp(-(dot(train\_in, nn\_weights))))

    # print("train\_out = \n", train\_out)

    mae = sum(abs(train\_out-train\_sol))/5

    min\_train\_loss = min(min\_train\_loss, mae)

    y.append(mae)

    nn\_weights += learning\_rate\*dot(train\_in.T, (train\_sol-train\_out)\*train\_out\*(1-train\_out))

  end = time.time()

  plt.plot(x, y, label='Train Loss')

  plt.legend(loc='upper right')

  plt.show()

  plt.close()

  print('min train loss:'+str(round(min\_train\_loss, 4)))

  print('it cost '+str(round(end-start, 4))+' sec to execute')

  return nn\_weights

epoch = 200

for i in range (4):

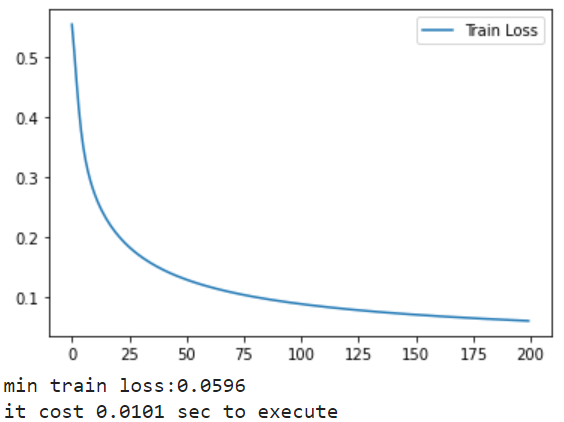
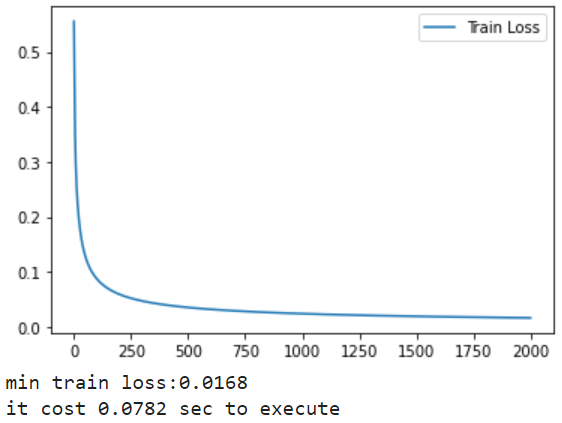
  epoch\_error(epoch, 0.1)

  epoch\_error(epoch, 1)

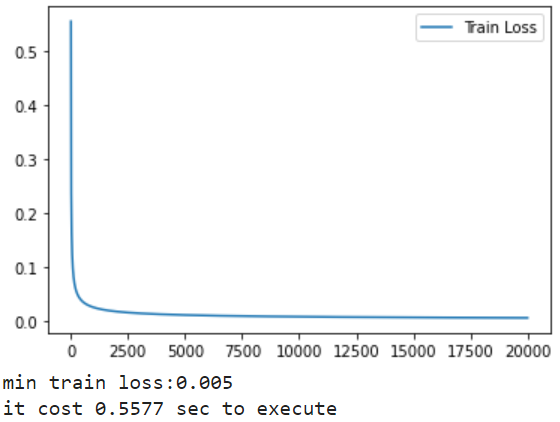
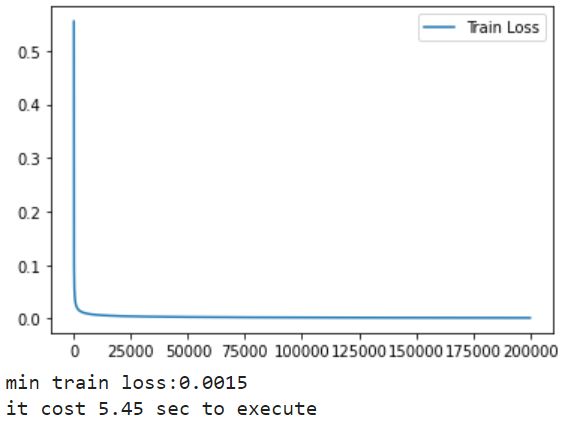
  epoch \*= 10

以下的誤差均以 Mean Absolute Error作評估，藍色線代表train\_out的loss。

以訓練不同次數做出誤差圖表，由以下圖來看，從左到右、從上到下，分別為訓練200、2000、20000、200000次的結果，可以看出從訓練200次到訓練200000次，其loss都在逐漸縮小，不過減緩的速度都在逐漸變慢。

訓練200次 訓練2000次

訓練20000次 訓練200000次

加入learning rate後，在不同次訓練次數中都可以看到，loss的減緩的速度變慢了，且在訓練200跟2000次時的訓練時間一些差距。

|  |  |  |
| --- | --- | --- |
|  | Learning rate=1 | Learning rate=0.1 |
| 訓練 200次 |  |  |
| 訓練 2000次 |  |  |
| 訓練 20000次 |  |  |
| 訓練200000次 |  |  |

1. Regularization（L1及L2），就你理解的，說明它是甚麼意思？請盡量用簡單的說法，讓外行人也能搞懂。

Regularization 泛指可以避免 overfitting 並增加模型泛化能力的方法

L1 regularization: 其主要是對各權重進行衰減，將一些不重要的特徵的權重為 0 ，留下模型認為重要的特徵。

L2 regularization: 其主要也是對各權重進行衰減但不會將權重更新為0。

1. TensorFlow Playground的Features有7種，請說明這7種形狀為何？在測試4種分類問題時用不用得著？哪些用得著？哪些用不著？Why？

這些Feature可以想成x, y座標平面上的性質

為該點x的值 為該點y的值

為該點x2的值 為該點y2的值

為該點x\*y的值

為該點sin(x)得值 為該點sin(y)的值

藍色為正值，黃色為負值

第一個分類問題的特徵為靠近中間的點皆較大的1，往外的點皆為較小的-1，所以有類似特徵的、、 、就派得上用場，其餘的特徵對這個分類問題就沒有太大的幫助。

第二個分類問題，其特徵就是在第一、三象限都是較小的 -1，二、四象限則是較大的1，所以有類似特徵的、、，其餘的特徵對這個分類問題就沒有太大的幫助。

第三個分類問題，其特徵就是在第二象限都是較小的-1，第四象限則是較大的1，所以有類似特徵的、就派得上用場，其餘的特徵對於這個分類問題就沒有太多的幫助。

第四個分類問題，其特徵比較不明顯，但是可以些微看出，其藍色與黃色都均勻的分佈在圖中，所以有類似特徵的、，就派得上用場，但只有這兩個特徵明顯無法組成這樣的pattern，所以我使用了Tensorflow Playground增加其他特徵並開啟6層每層皆有8個layer的hidden layer，以L1正規化，透過模型訓練的權重讓我知道還有哪些特徵是重要的，在實驗過後，找出還有兩個特徵、，對於這個分類問題也有很大的幫助，其餘的特徵對於這個分類問題就沒有太多的幫助。

1. 迴歸（Regression）老師講得不明不白。請你自己設法了解清楚，說明Regression為何？有何用處？

假設x, y之間存在一個關係f，使得f(x)=y，而Regression是找一個函式h(x)使h(x)盡可能接近f(x)，通常應用於以時間做變數的問題，比如說天氣預測的問題，想知道今天平均溫度，可以將這個問題定義為f，x為時間，而透過Regression我們找出一個函數h(x)，可以輸入今天的時間x，預測出今天平均溫度。

1. TensorFlow Playground的Activation激活函數有4種，請說明這4種形狀為何？在測試時用不用得著？哪些用得著？哪些用不著？Why？

TensorFlow Playground的激活函數畫出來長的如下，一旁是其定義。

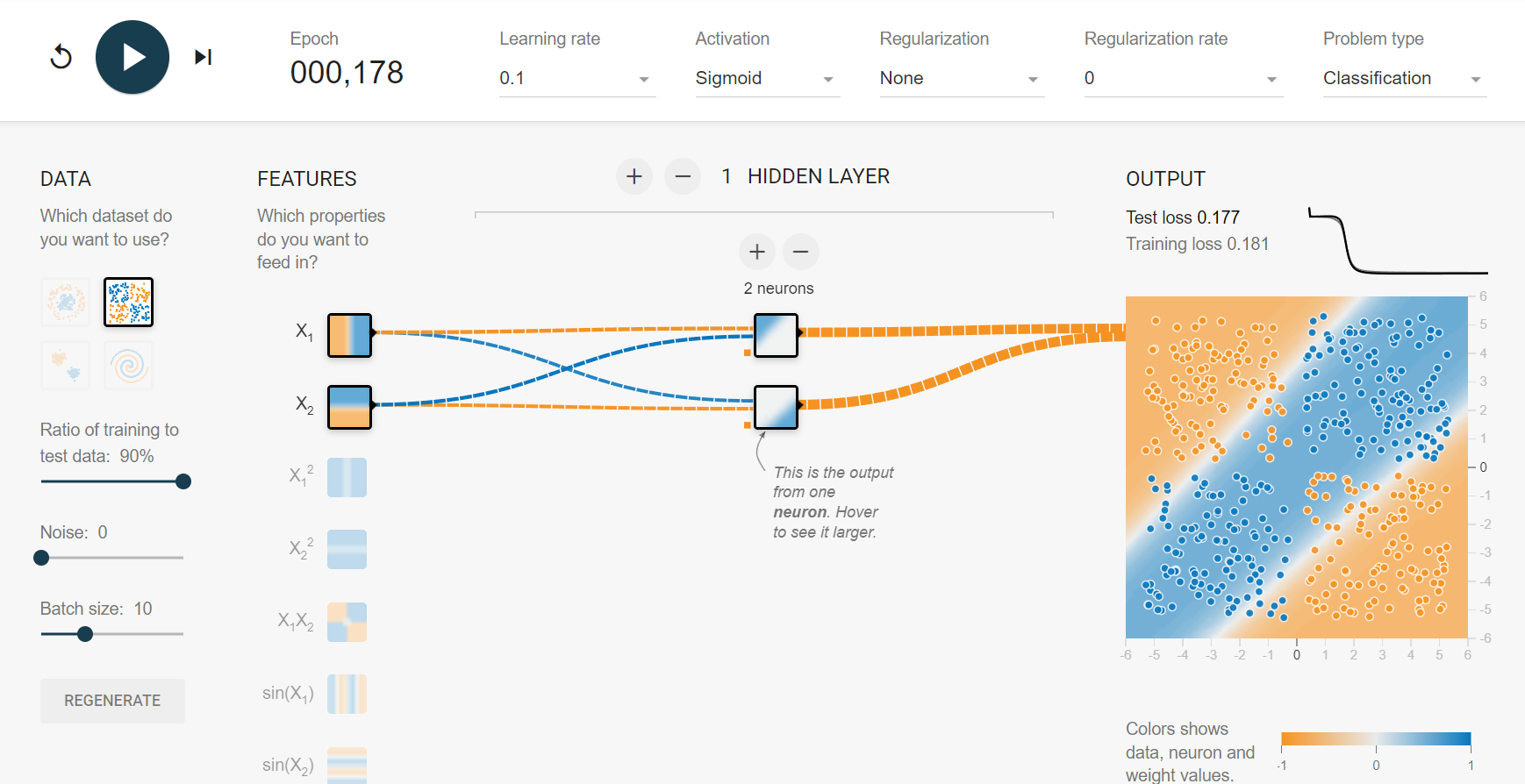
第一、二、三個分類問題中因為其分類可以以各特徵的非線性或線性變換得到，所以四種激勵函數都用的著，他們都可以達到使最終的結果收斂。

第四個分類問題由於不是單純線性轉換可以得到的結果，所以需要靠非線性的激勵函式ReLU, Tanh, Sigmoid，單純做線性轉換的Linear就沒有任何幫助。

1. 接著你想自己訓練一下講義上第4頁的「Solving XOR with a Neural Net」問題，並用一樣多的layers及neurons，你能做出雷同的結果嗎？說明你如何執行訓練及測試？如何得出正確率？你自己是怎麼算的？這個訓練結果花了多少時間？能做到令人滿意的結果嗎？

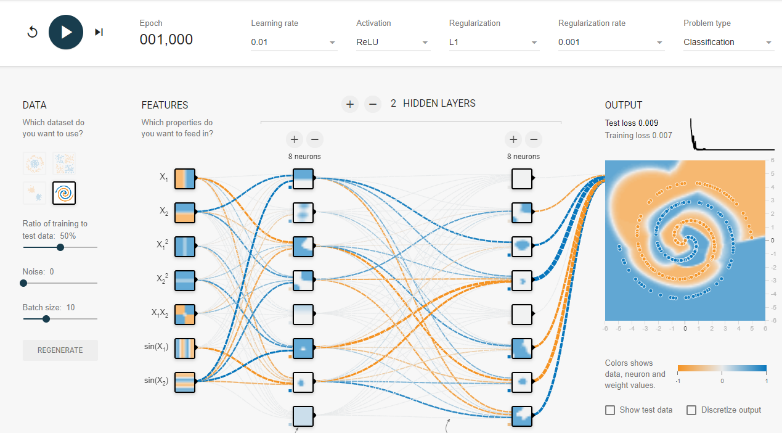
這邊以TensorFlow playground 的第二個分類問題作訓練，接著對這個問題做訓練，以下圖中參數做訓練

(Learning rate=0.1, Activate function=Sigmoid, Regularization=None )，結果出來之後再跟上圖做比照，可以看出做出了類似的結果，正確率計算這邊以分類正確/總數量做為正確率，計算其是否分類正確，從右下圖可以清楚看到，分類正確的有391個，分類錯誤的有57個，所以正確率為87%，整個實驗過程花了178個epoch，訓練3秒。

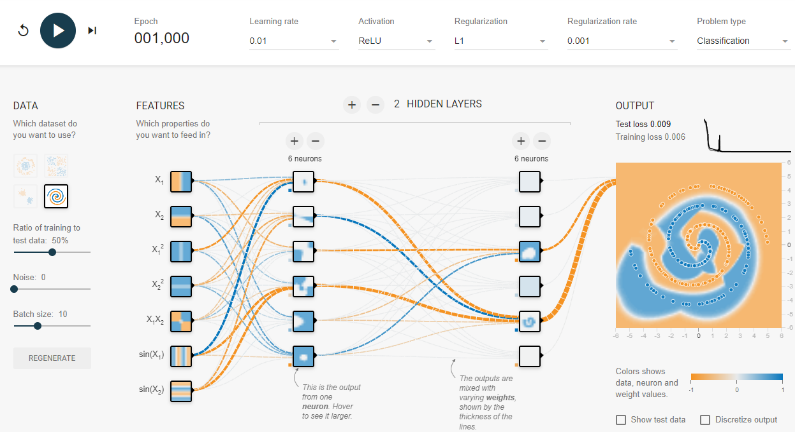


1. 接著你想自己訓練一下複雜的螺旋狀「分類」問題，並盡可能用越少的layers及neurons越好，說明你如何執行訓練及測試？如何得出正確率？你自己是怎麼算的？這個訓練結果花了多少時間？能做到令人滿意的結果嗎？

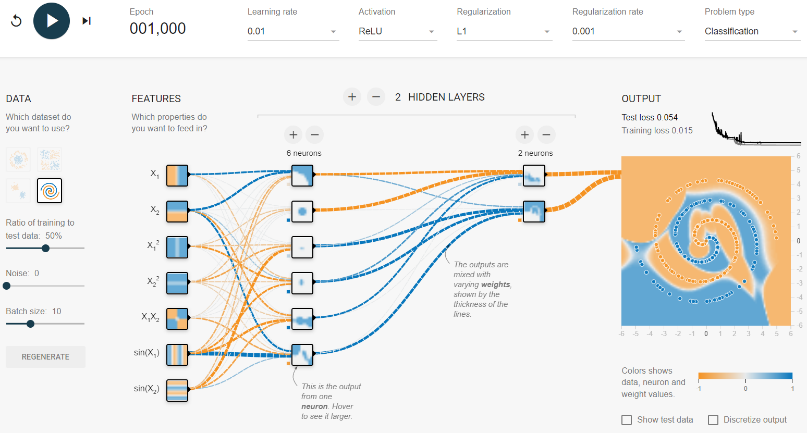
以下訓練皆以ReLU作為Activation function，並以低於0.01的loss作為滿意的結果的目標。

首先先以兩層的hidden layer跟各8個neuron的架構進行訓練，並且使用L1正規化使權重衰減，可以看到這一次的訓練就已經達成了loss低於0.01的目標，我們觀察各個neuron輸出的weight，我決定先減少輸出權重皆為為0的neuron

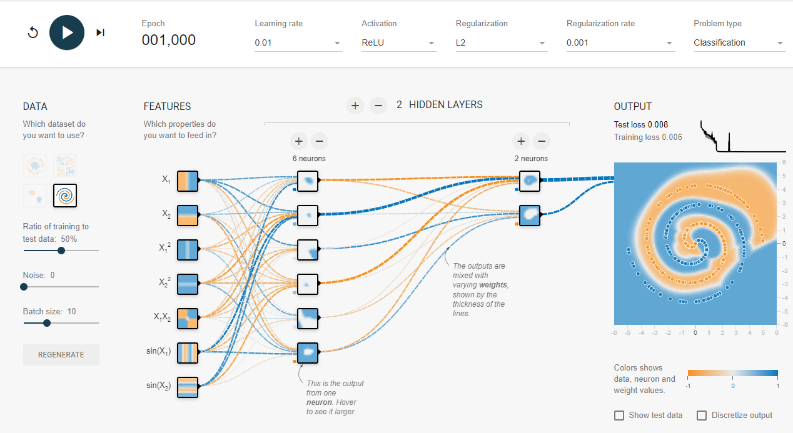
（訓練時間：34秒）

第二次作訓練，在減少neuron的情況也達到了一樣低的loss，而且第二層又出現了四個輸出權重皆為0的neuron，這邊一樣將其去除。

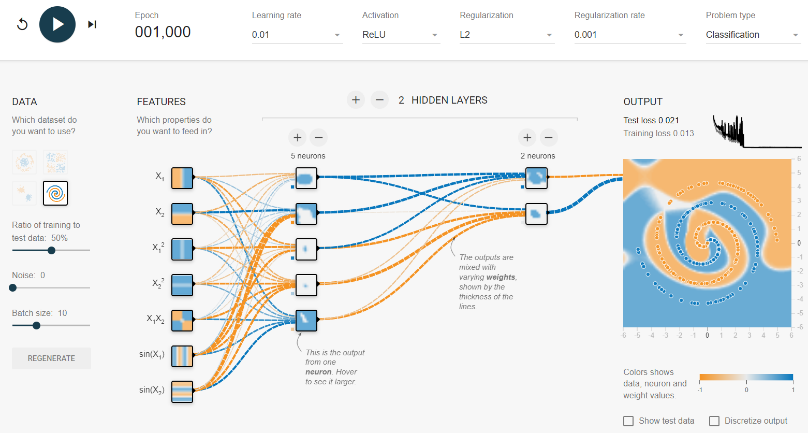
（訓練時間：26秒）

第三次訓練，可以發現loss開始變高，所以減少neuron的步驟暫時結束，這邊改以L2正規進下一步訓練，看是否可以維持跟第二次一樣低的loss。

（訓練時間：21秒）

第四次訓練使用L2正規化後，loss成功降低到0.01以下了，大部分的neuron都有一定的權重輸出，所以這邊減少一個輸出權重最低的neuron作實驗，看能不能達到一樣低的loss。

（訓練時間：22秒）

第五次訓練可以看出減少一個neuron後loss就沒辦法降低到0.01以下了。

（訓練時間：22秒）

所以最後得出結果，第四次的結果最好

要達到loss小於0.01的結果至少要

兩層hidden layer且分別有6個跟2個neuron

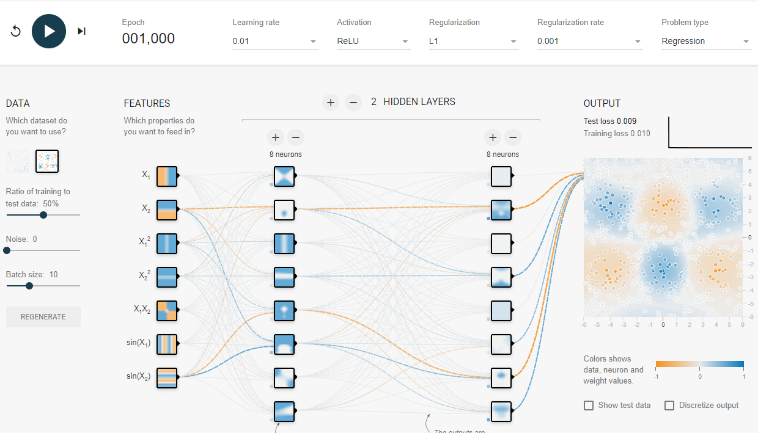
訓練時間：22秒

最後正確率因不清楚loss是如何算出的，所以一個一個點數並以(分類正確的點的數量/總點數)計算181個點中有179個點在正確的分類中，所以正確率為98.9%。

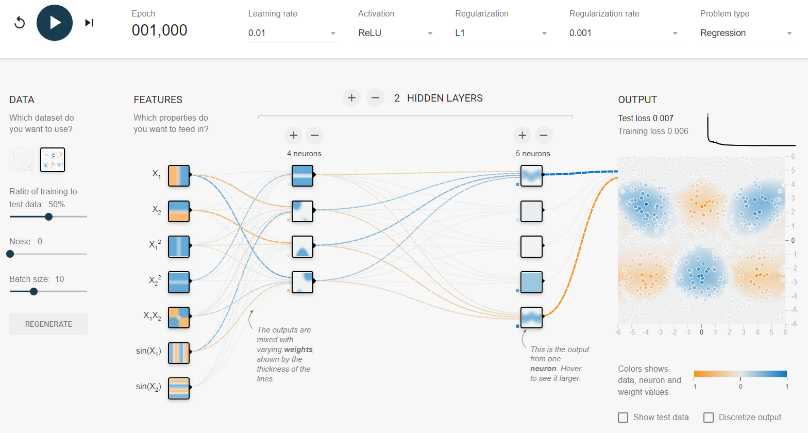
1. 接著你想自己訓練一下一些複雜的第二個「迴歸」問題，並盡可能用越少的layers及neurons越好，說明你如何執行訓練及測試？如何得出正確率？你

自己是怎麼算的？這個訓練結果花了多少時間？能做到令人滿意的結果嗎？

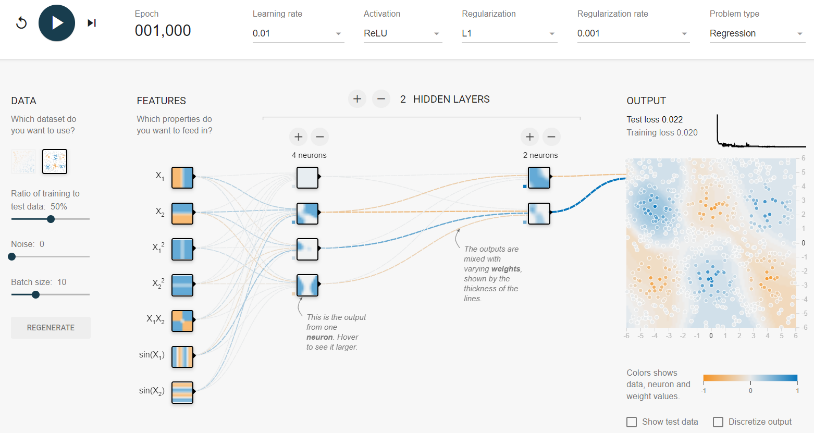
以下訓練皆以ReLU作為Activation function，以低於0.01的loss作為滿意的結果的目標。

第一次訓練採用兩層的hidden layer跟各8個neuron的架構進行訓練，並且使用L1正規化使權重衰減，可以看到這一次的訓練就已經達成了loss低於0.01的目標，觀察各個neuron輸出的weight，我決定先刪除輸出權重皆為為0的neuron

（訓練時間：32秒）

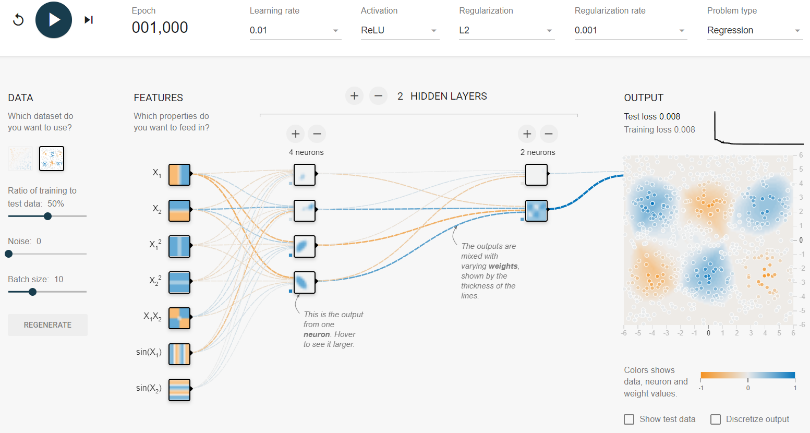
第二次訓練及使減少了neuron的數量也不會讓loss變多，在這次訓練中甚至變更低了，下一步是再次刪除輸出權重為0的neuron。

（訓練時間：21秒）

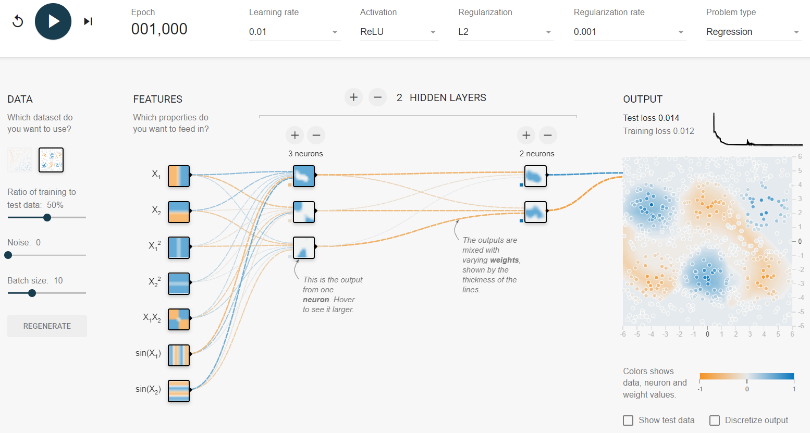


第三次訓練，發現loss開始變高，所以決定下一步將使用L2正規化

（訓練時間：20秒）

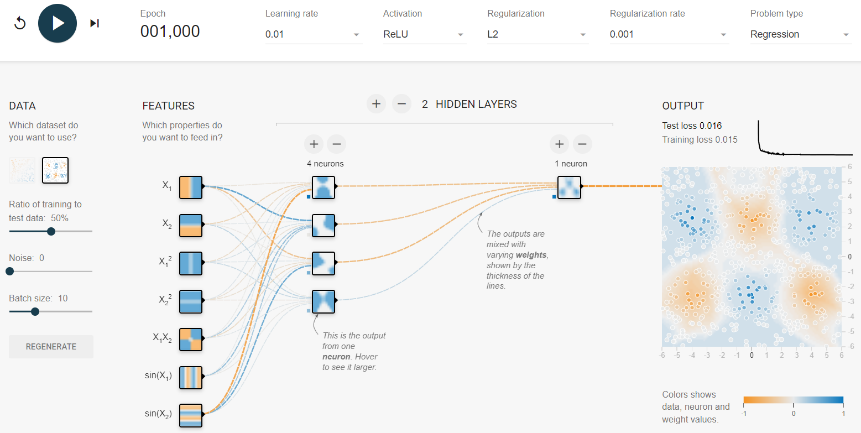
第四次訓練使用了L2正規化後loss成功減少到0.008

（訓練時間：20秒）

第五第六次分別將第四次架構中的一層跟的二層拿掉一個neuron看使否能達到依樣低的loss，但實驗結果顯示少了兩個其中一個都會讓loss超過0.01

（訓練時間：18秒）

拿掉第一層的一個neuron



（訓練時間：17秒）

拿掉第二層的一個neuron

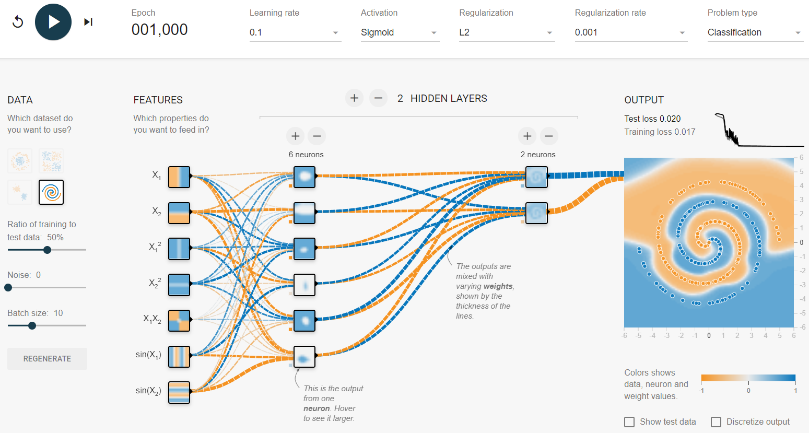
所以最後得出結果，要達到loss小於0.01的結果至少要

兩層hidden layer且分別有6個跟2個neuron

訓練時間為20秒

最後正確率部分因為找不到regression問題的正確率計算方法，所以無法計算。

1. 上兩小題使用一種activation function(如sigmoid)，若改用其他的functions（至少請舉2種其他的functions）來跑程式，請問結果有何不同？訓練速度有差別嗎？



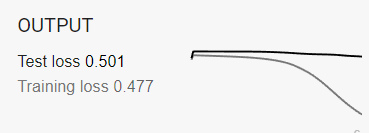
第八題如改用Sigmoid進行測試，在一樣1000次的epoch中實驗結果如下。

（最低loss： 0.02，較ReLU高出0.012）

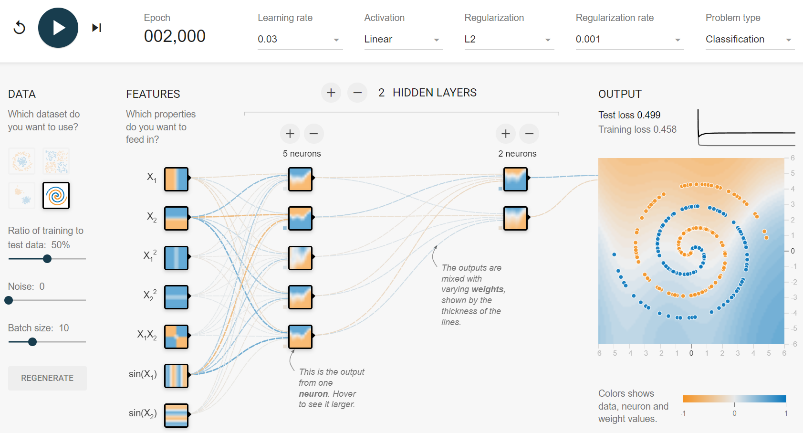
（訓練時間：23秒，較ReLU多1秒）

可以看出loss部分相差有點大，但訓練時間差不多

且loss降低的速度從下圖可以看出ReLU降低的速度較快。

使用Sigmoid前100個epoch的loss變化 使用ReLU前100個epoch的loss變化

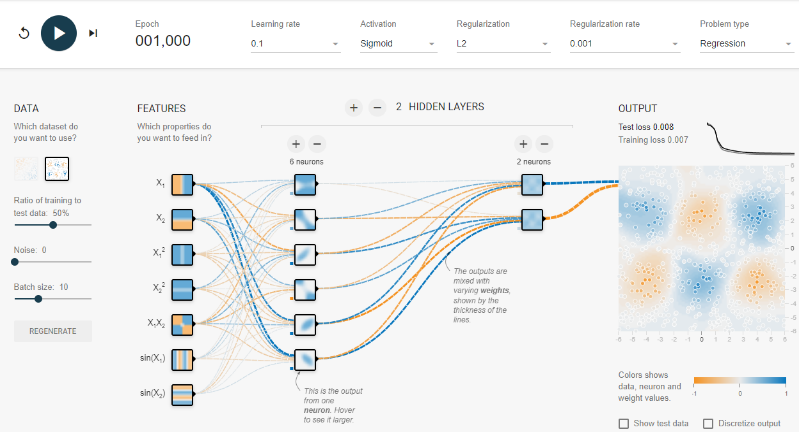
如改用Linear，一樣在其他條件皆不改變的情況實驗結果如下。

（最低loss：0.499，較ReLU高0.491）

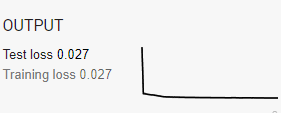
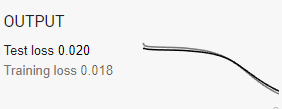
（訓練時間：18秒，較ReLU少4秒。）

可以看出雖然訓練時間少了18%，但是loss卻高了0.491

所以可以判定這個Activation function對於這個分類問題沒有幫助。

第九題如改用Sigmoid進行測試，在一樣1000次的epoch中實驗結果如下

且loss降低的速度從下圖可以看出ReLU降低的速度較快。

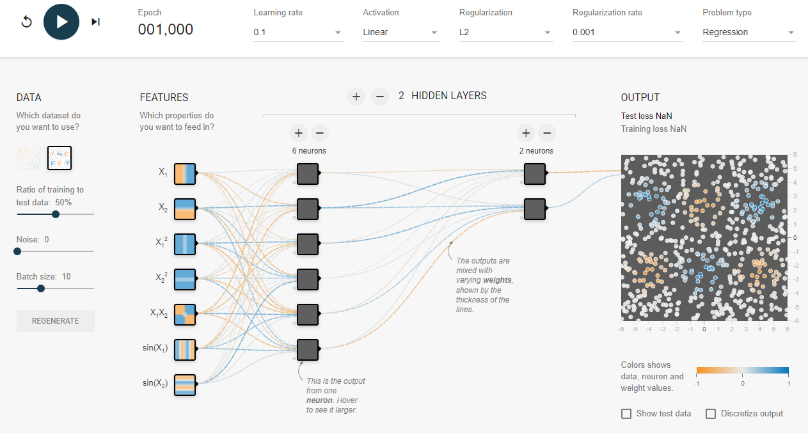
 

使用Sigmoid前100個epoch的loss變化 使用ReLU前100個epoch的loss變化

（最低loss：0.008，與ReLU相同）

（訓練時間為27秒，較使用ReLU多7秒。）

可以看出雖然loss一樣，但是訓練時間多出了35%

如改用Linear，一樣在其他條件皆不改變的情況實驗結果如下

（最低loss：NaN）

（訓練時間：20秒，與使用ReLU相同）

可以明顯看出，使用Linear作為Activation function，根本無法訓練出東西。

1. 請說明你作此作業所碰到的一些狀況及困難。

對於許多人工智慧中常用到的名詞，像是Activation function、Features、Regularization…都還不清楚，在查詢一些相關定義的時候常常會出現更多不清楚的名詞，就要繼續接著先搞清楚那個看不懂的名詞，再回來看原本那個詞的定義，或是在訓練Tensorflow Playground時看到loss的曲線開始來回震盪時，會不知道是哪裡出了問題，這些東西也都需要一個一個去查，將這些未知的名詞都整理過後才能知道這些名詞的用處，調整了某個參數會讓模型如何收斂等。

1. 請列出你的參考文獻（含網站）來源，並請說明參考了哪些部分用於作業中。

Regression相關定義：

[https://p61402.github.io/2019/06/12/機器學習經典演算法實作-Linear-Regression/](https://p61402.github.io/2019/06/12/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E7%B6%93%E5%85%B8%E6%BC%94%E7%AE%97%E6%B3%95%E5%AF%A6%E4%BD%9C-Linear-Regression/)

Activation function相關圖片：

<https://en.wikipedia.org/wiki/Activation_function>

Activation function相關解釋：

<https://codingnote.cc/zh-tw/p/176736/>