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

2. 「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）

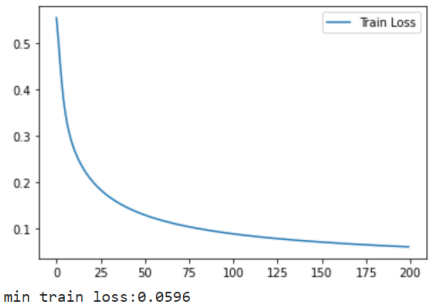
   程式部分

```
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
```

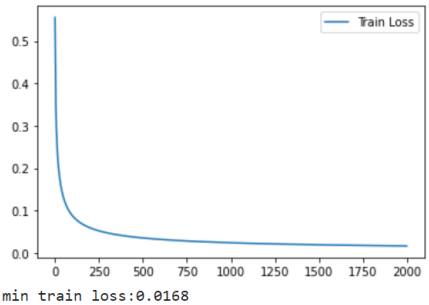
以下的誤差均以 Mean Absolute Error 作評估，藍色線代表 train_out 的 loss。

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。



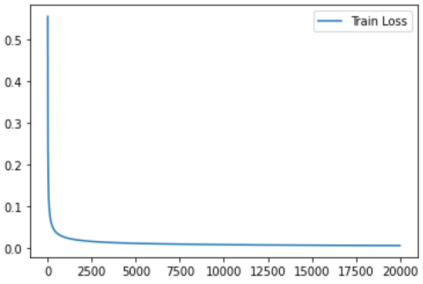

```
min train loss:0.0596
it cost 0.0101 sec to execute
```

訓練 200 次



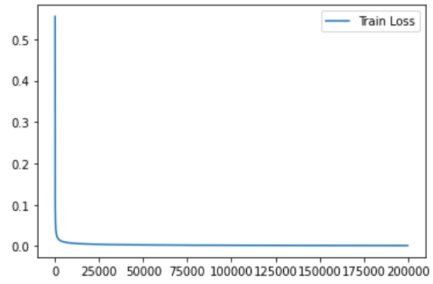

```
min train loss:0.0168
it cost 0.0782 sec to execute
```

訓練 2000 次




```
min train loss:0.005
it cost 0.5577 sec to execute
```

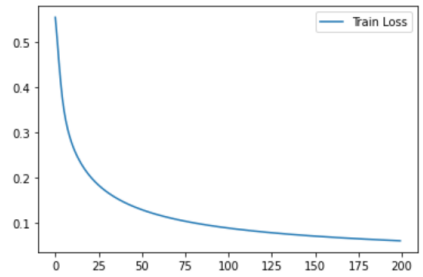
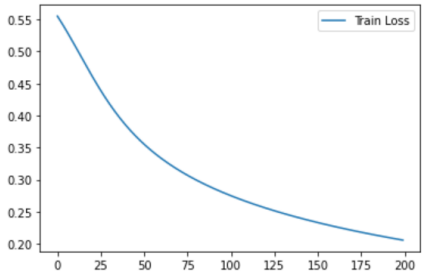
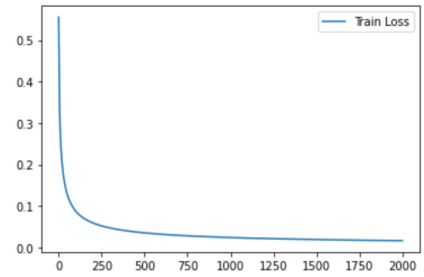
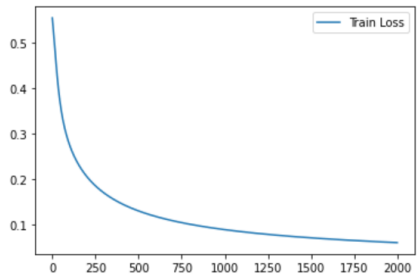
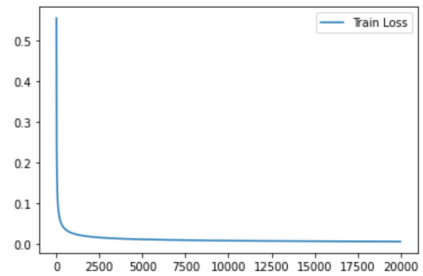
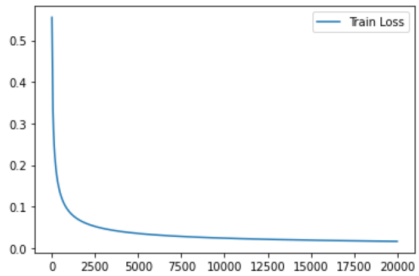
訓練 20000 次

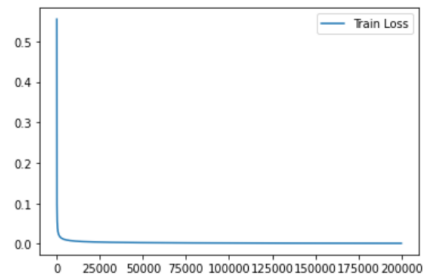
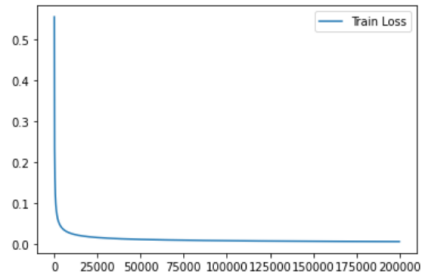



```
min train loss:0.0015
it cost 5.45 sec to execute
```

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	 min train loss:0.0596 it cost 0.0101 sec to execute	 min train loss:0.2057 it cost 0.0048 sec to execute
訓練 2000 次	 min train loss:0.0168 it cost 0.0782 sec to execute	 min train loss:0.0597 it cost 0.0581 sec to execute
訓練 20000 次	 min train loss:0.005 it cost 0.5577 sec to execute	 min train loss:0.0168 it cost 0.56 sec to execute

訓練 200000 次	 min train loss:0.0015 it cost 5.45 sec to execute	 min train loss:0.005 it cost 5.4432 sec to execute

3. 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。

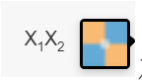
   L2 regularization: 其主要也是對各權重進行衰減但不會將權重更新為 0。

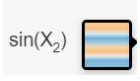
4. TensorFlow Playground 的 Features 有 7 種，請說明這 7 種形狀為何？在測試 4 種分類問題時用不用得著？哪些用得著？哪些用不著？Why？

   這些 Feature 可以想成 x, y 座標平面上的性質

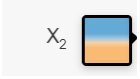
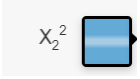
    為該點 x 的值　  為該點 y 的值

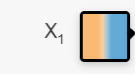
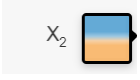
    為該點 x^2 的值　  為該點 y^2 的值

    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。

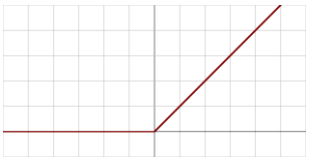
 第四個分類問題，其特徵比較不明顯，但是可以些微看出，其藍色與黃色都均勻的分佈在圖中，所以有類似特徵的  、  ，就派得上用場，但只有這兩個特徵明顯無法組成這樣的 pattern，所以我使用了 Tensorflow Playground 增加其他特徵並開啟 6 層每層皆有 8 個 layer 的 hidden layer，以 L1 正規化，透過模型訓練的權重讓我知道還有哪些特徵是重要的，在實驗過後，找出還有兩個特徵  、  ，對於這個分類問題也有很大的幫助，其餘的特徵對於這個分類問題就沒有太多的幫助。

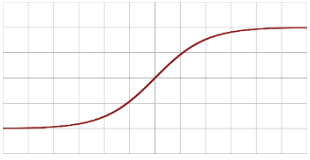
5. 迴歸（Regression）老師講得不明不白。請你自己設法了解清楚，說明 Regression 為何？有何用處？

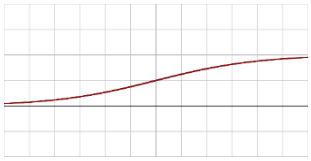
   假設 x, y 之間存在一個關係 f，使得 f(x)=y，而 Regression 是找一個函式 h(x)使 h(x)盡可能接近 f(x)，通常應用於以時間做變數的問題，比如說天氣預測的問題，想知道今天平均溫度，可以將這個問題定義為 f，x 為時間，而透過 Regression 我們找出一個函數 h(x)，可以輸入今天的時間 x，預測出今天平均溫度。

6. 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。

   ReLU  $f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$

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

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

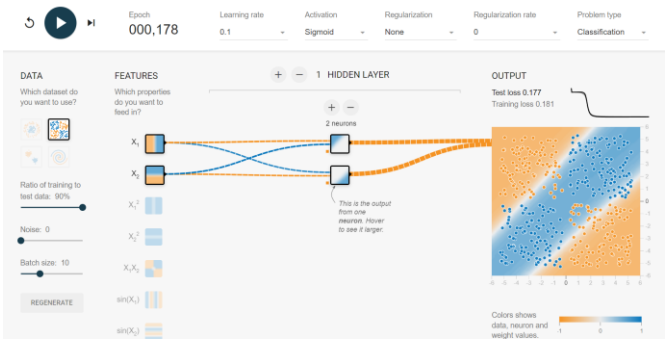
Linear  $f(x) = x$

 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。

7. 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？

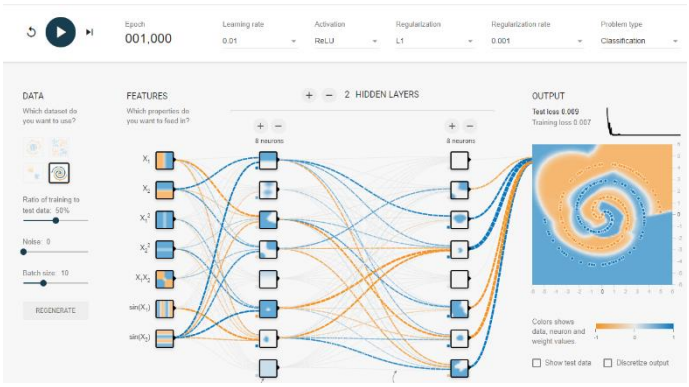
   這邊以 TensorFlow playground 的第二個  分類問題作訓練，接著對這個問題做訓練，以下圖中參數做訓練
   (Learning rate=0.1, Activate function=Sigmoid, Regularization=None)，結果出來之後再跟上圖做比照，可以看出做出了類似的結果，正確率計算這邊以分類正確/總數量做為正確率，計算其是否分類正確，從右下圖可以清楚看到，分類正確的有 391 個，分類錯誤的有 57 個，所以正確率為 87%，整個實驗過程花了 178 個 epoch，訓練 3 秒。



8. 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

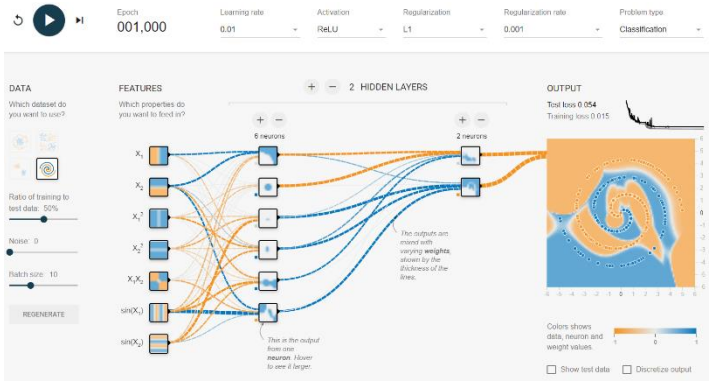


（訓練時間：34 秒）

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。

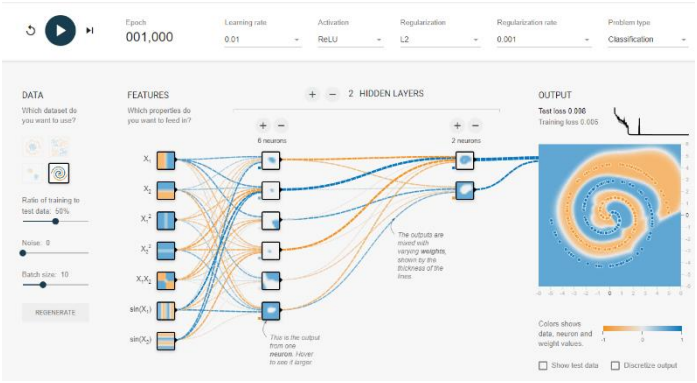
（訓練時間：26 秒）

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。



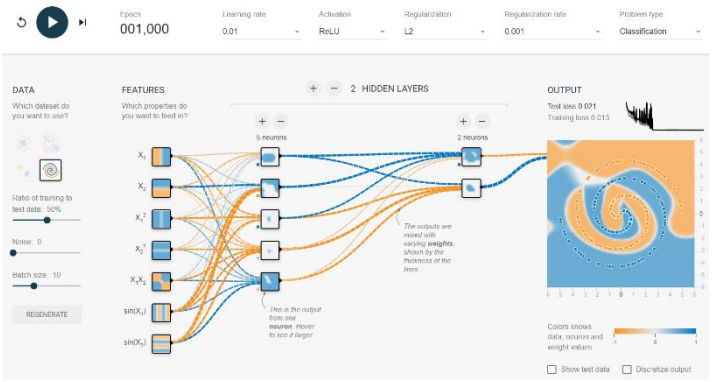
（訓練時間：21 秒）

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。



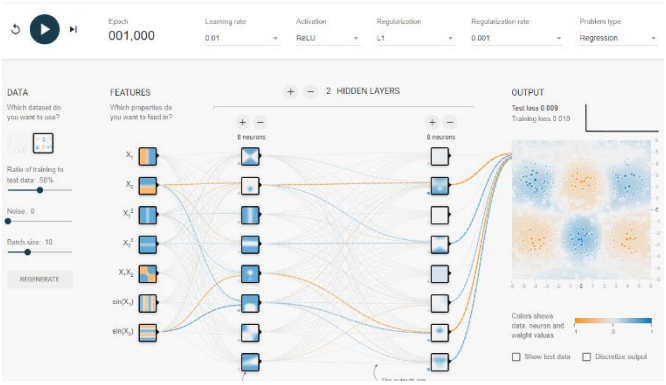
（訓練時間：22 秒）

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%。

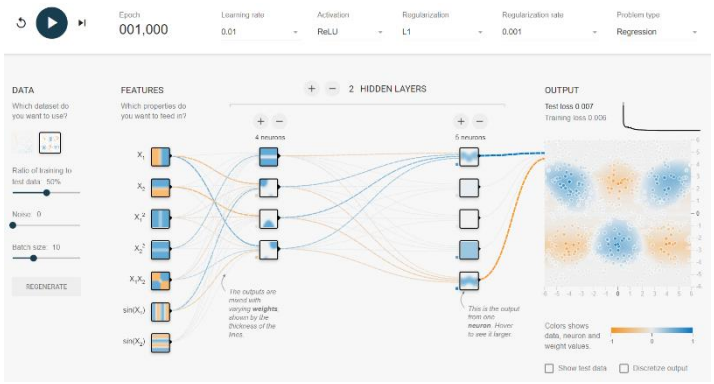
9. 接著你想自己訓練一下一些複雜的第二個「迴歸」問題，並盡可能用越少的 layers 及 neurons 越好，說明你如何執行訓練及測試？如何得出正確率？你

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）

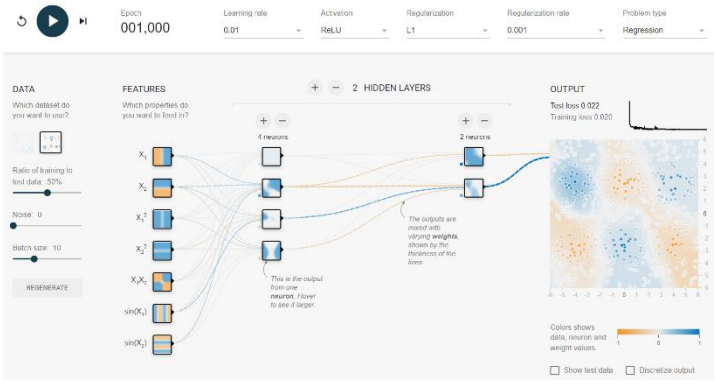


第二次訓練及使減少了 neuron 的數量也不會讓 loss 變多，在這次訓練中甚至變更低了，下一步是再次刪除輸出權重為 0 的 neuron。



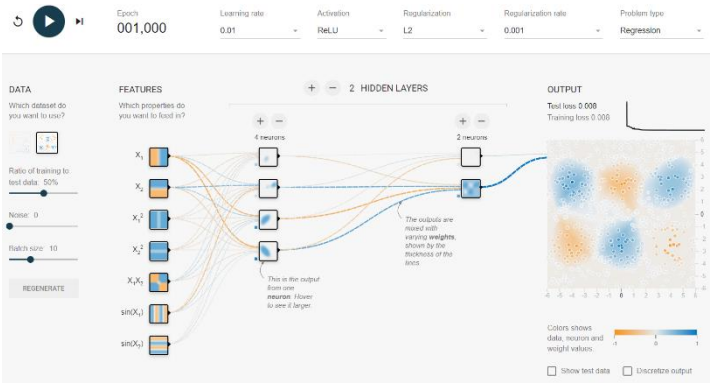
（訓練時間：21 秒）

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



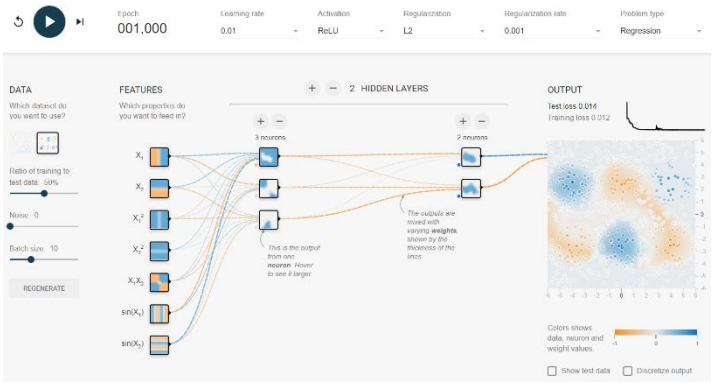
（訓練時間：20 秒）

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

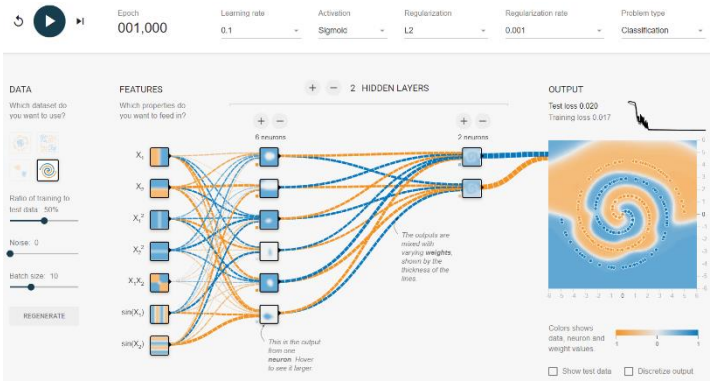
（訓練時間：17 秒）

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。

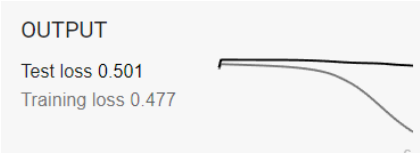
10. 上兩小題使用一種 activation function(如 sigmoid)，若改用其他的 functions（至少請舉 2 種其他的 functions）來跑程式，請問結果有何不同？訓練速度有差別嗎？

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

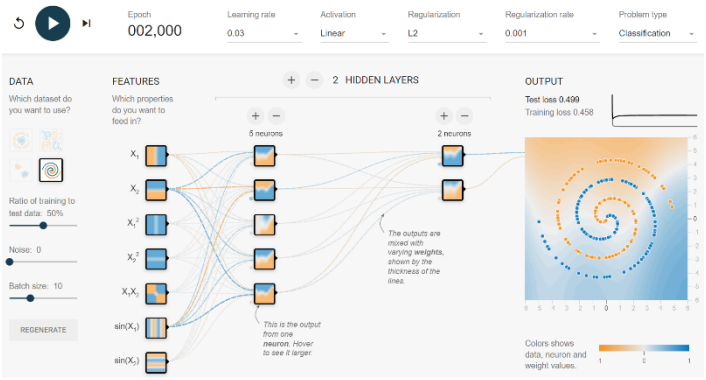
且 loss 降低的速度從下圖可以看出 ReLU 降低的速度較快。



使用 Sigmoid 前 100 個 epoch 的 loss 變化

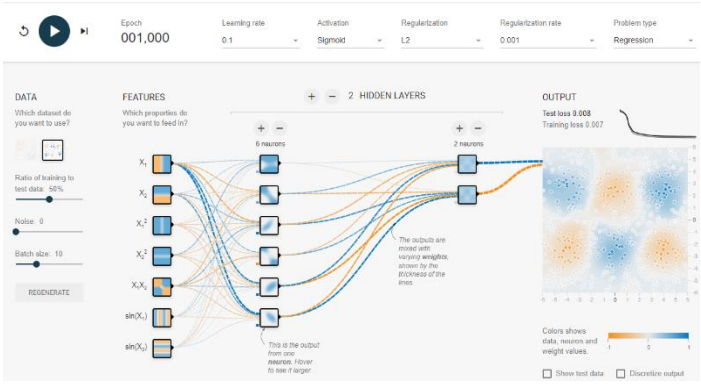
使用 ReLU 前 100 個 epoch 的 loss 變化

如改用 Linear，一樣在其他條件皆不改變的情況實驗結果如下。

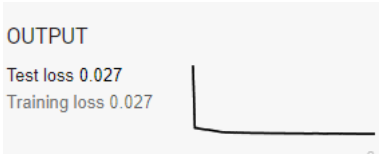


（最低 loss：0.499，較 ReLU 高 0.491）
（訓練時間：18 秒，較 ReLU 少 4 秒。）
可以看出雖然訓練時間少了 18%，但是 loss 卻高了 0.491
所以可以判定這個 Activation function 對於這個分類問題沒有幫助。

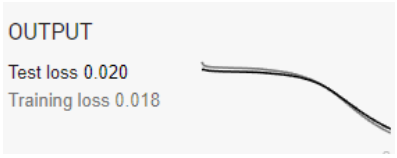
第九題如改用 Sigmoid 進行測試，在一樣 1000 次的 epoch 中實驗結果如下



且 loss 降低的速度從下圖可以看出 ReLU 降低的速度較快。



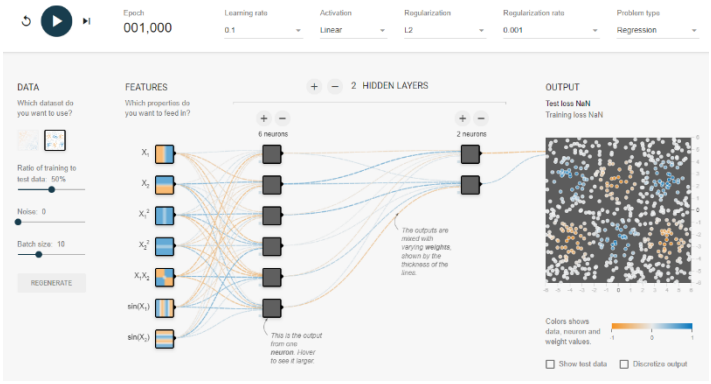
使用 Sigmoid 前 100 個 epoch 的 loss 變化



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。

11. 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。

12. 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/

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/