AI HW1

姓名：許哲安
學號：40871107H

1.

macOS Big Sur Version 11.6.4
MacBook Pro (15-inch, 2019)
Processor: 2.3 GHz 8-Core Intel Core i9
Memory: 16 GB 2400 MHz DDR4
Graphics: Radeon Pro 560X 4 GB
Intel UHD Graphics 630 1536 MB
電話：0905235578
當初選擇這台電腦是因為剛好開學，學校旁的電腦店有配合開學優惠，平常對於蘋果產品較熟悉，也剛好有特價，所以就購買了。

2.

(1)會，根據自己的實驗情況，將訓練次數調整為 625000 次，並在訓練次數為 200, 1000, 5000, 25000, 125000, 625000 時，印出誤差，計算誤差的方式為 train_out 與 train_sol 的差，再取絕對值，以下為誤差圖表，column 是訓練次數，row 是 train data

誤差表

	200	1000	5000	25000	125000	625000
Train Data 1	6.10268E-02	2.768871E-02	1.233986E-02	5.49879E-03	2.45411E-03	1.09639859E-03
Train Data 2	1.0103035E-01	4.356035E-02	1.907765E-02	8.44302E-03	3.75721E-03	1.67645994E-03
Train Data 3	2.785303E-02	7.62706E-03	2.1692E-03	6.3276E-04	1.8702E-04	5.56314209E-05
Train Data 4	2.290194E-02	6.37242E-03	1.82409E-03	5.3358E-04	1.5791E-04	4.69958271E-05
Train Data 5	8.419804E-02	3.661281E-02	1.608587E-02	7.12836E-03	3.17399E-03	1.41658166E-03

(2)會，根據網路上的資料與我的理解，learning rate 是針對在調整 weights 時，決定調整幅度多大的一個值，訓練類神經網路的目標是找到一組 weights 使得誤差為最小值，假如 learning rate 太大，可能一不小心調整過頭 ; learning rate 太小，可能調整幅度過小，造成訓練次數需提高，才能得到最小誤差值。
實驗設計：分別在 learning rate 為 0.00001、1、10 的情況下，訓練次數為 625000，並在訓練次數為 200, 1000, 5000, 25000, 125000, 625000 時，印出誤差（誤差計算方式與上小題相同），learning rate 0.00001 與 10 是參考 tensorflow playground learning rate 的最小值與最大值，而 1 是我自己設定的。

Learning Rate=0.00001誤差表

	200	1000	5000	25000	125000	625000
Train Data 1	7.0690138E-01	7.0614152E-01	7.023215E-01	6.8273819E-01	5.7845576E-01	2.73353E-01
Train Data 2	2.6904506E-01	2.6927989E-01	2.7045868E-01	2.7646223E-01	3.0813907E-01	3.9114903E-01
Train Data 3	3.6371362E-01	3.6356585E-01	3.6282862E-01	3.5918886E-01	3.4285094E-01	2.8265633E-01
Train Data 4	7.6221404E-01	7.6158196E-01	7.5839865E-01	7.4191838E-01	6.477534E-01	2.9306884E-01
Train Data 5	6.7363356E-01	6.7327026E-01	6.714458E-01	6.6212341E-01	6.1086743E-01	4.0331117E-01

Learning Rate=1 誤差表

	200	1000	5000	25000	125000	625000
Train Data 1	6.10268E-02	2.768871E-02	1.233986E-02	5.49879E-03	2.45411E-03	1.09639859E-03
Train Data 2	1.0103035E-01	4.356035E-02	1.907765E-02	8.44302E-03	3.75721E-03	1.67645994E-03
Train Data 3	2.785303E-02	7.62706E-03	2.1692E-03	6.3276E-04	1.8702E-04	5.56314209E-05
Train Data 4	2.290194E-02	6.37242E-03	1.82409E-03	5.3358E-04	1.5791E-04	4.69958271E-05
Train Data 5	8.419804E-02	3.661281E-02	1.608587E-02	7.12836E-03	3.17399E-03	1.41658166E-03

Learning Rate=10誤差表

	200	1000	5000	25000	125000	625000
Train Data 1	1.929295E-02	8.68332E-03	3.88145E-03	1.73413123E-03	7.75051170E-04	3.4650484E-04
Train Data 2	3.005253E-02	1.33745E-02	5.95047E-03	2.65315675E-03	1.18474983E-03	5.29462279E-04
Train Data 3	4.32692E-03	1.2671E-03	3.7353E-04	1.10862945E-04	3.30339645E-05	9.86258267E-06
Train Data 4	3.62809E-03	1.06711E-03	3.1519E-04	9.36293496E-05	2.79098335E-05	8.33420255E-06
Train Data 5	2.530437E-02	1.128511E-02	5.02545E-03	2.24161070E-03	1.00115380E-03	4.47448844E-04

實驗結果：發現 learning rate 越大，調整的幅度越大，以 Train Data 1 為例，learning rate 為 0.00001 時，訓練 1000 次的誤差比 200 次少約 0.008;learning rate 為 1 時，訓練 1000 次的誤差比 200 次少約 3.4 ; learning rate 為 10 時，訓練 1000 次的誤差比 200 次少約 11，在多 800 次的訓練下，learning rate 越大，調整幅度越大，誤差越小。另外發現一個比較出乎預料的點，在 learning rate 為 0.00001 的 Train Data 2，隨著訓練次數增加，誤差竟然越來越大 。

程式碼：

```
from numpy import *

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

# initialize nn_weights
random.seed(1)
```

```
nn_weights = 2 * random.random((3, 1)) - 1

# train the network
for i in range(625001):

  # Calculate the outputs for each training examples
  train_out = 1 / (1+exp(-(dot(train_in, nn_weights))))


  if i in [200, 1000, 5000, 25000, 125000, 625000]:
    error = absolute(train_sol - train_out)
    print(f"error with trained cases {i} = ")
    print(error)

  # Run the NN adjustments
  # LR means Learning Rate


  LR = 0.00001
  nn_weights += dot(train_in.T, (train_sol-train_out)*train_out*(1-train_out)) * LR

# unknown test input
test_in = array([1, 0, 0])

# print the result for our unknown test input
print("\nThe final prediction is ", 1/(1+exp(-(dot(test_in, nn_weights)))))
```

3.

在訓練類神經網路時，有種現象叫做 overfitting，意指在訓練過程中表現很好（誤差小），但在測試時卻表現很差（誤差大），可以說是把簡單的問題複雜化，產生過多不必要的參數，造成 overfitting 的原因可能是訓練資料過少、feature 選擇過多…等，而 Regularization 的動作是幫助 overfitting 的模型，退回到較簡單的模型。
Regularization 的方式有兩種

(1) L1 (Lasso)
主要是修改模型中 weight 的值，將沒有用的 feature 的 weight 設為 0，留下較重要的權重，換句話說，透過 L1 Regularization，可以減少 feature 的種類，使得原本複雜的模型(feature 多)簡單化（留下重點 feature，捨去相較不太重要的 feature）。

(2) L2 (Ridge)

也是修改模型中 weight 的值，但與 L1 不同的地方是， L2 不會將沒有用的 feature 其 weight 設為 0，而是保留所有的 weight 且降低 weight 的值，weight 值越大，降低越多，換句話說是削弱所有 weight，但依然保留用處較少的 feature。

4.

大於 0 為藍色，小於 0 為橘色，接近 0 為白色。

(1)X1

X1 是以 x 軸做區分，中間白色部分為 x = 0 的線，於此線右方為 x>0 的部分，顯示為藍色，左方為 x<0 的部分，顯示為橘色。

(2)X2

X2 是以 y 軸為區分，中間白色部分為 y = 0 的線，於此線上方為 y>0 的部分，顯示為藍色，下方為 y<0 的部分，顯示為橘色。

(3)X1^2

X1^2 為將 X1 的圖乘上 X1，中間白色部分為 x^2 = 0 的線，於此線的左右方都為 x^2>0 的部分，顯示為藍色，在 X1 圖中中間白線的右方為正，正數乘積依然是正數，中間白線左方為負，負數乘積為正，所以說明 X1^2 左右兩方都大於 0，顯示為藍色。

(4)X2^2

X2^2 為將 X2 的圖乘上 X2，中間白色部分為 y^2 = 0 的線，於此線的上下方都為 y^2>0 的部分，顯示為藍色，在 X2 圖中中間白線的上方為正，正數乘積依然是正數，中間白線下方為負，負數乘積為正，所以說明 X2^2 上下兩方都大於 0，顯示為藍色。

(5)X1*X2

X1*X2 為將 X1 的圖乘上 X2，中間白色部分為 x*y = 0 的部分，假設此圖為一個 xy 座標，第一象限為 X1 與 X2 都是正數，乘積為正，顯示為藍色，第二象限為 X1 是負數，X2 是正數，乘積為負，顯示為橘色，第三象限為 X1 與 X2 都是負數，乘積為正，顯示為藍色，第四象限為 X1 是正數，X2 是負數，乘積為負，顯示為橘色。

(6)sin(X1)

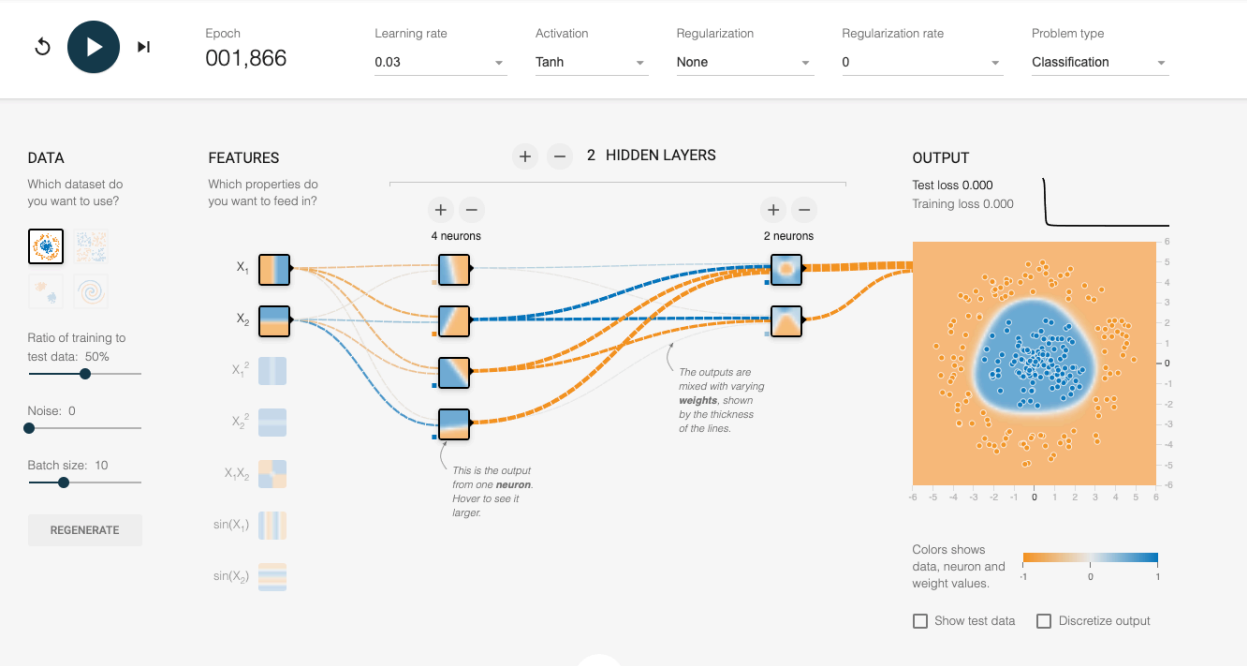
Sin(X1)為 y=sin(x)的值

(7)sin(X2)

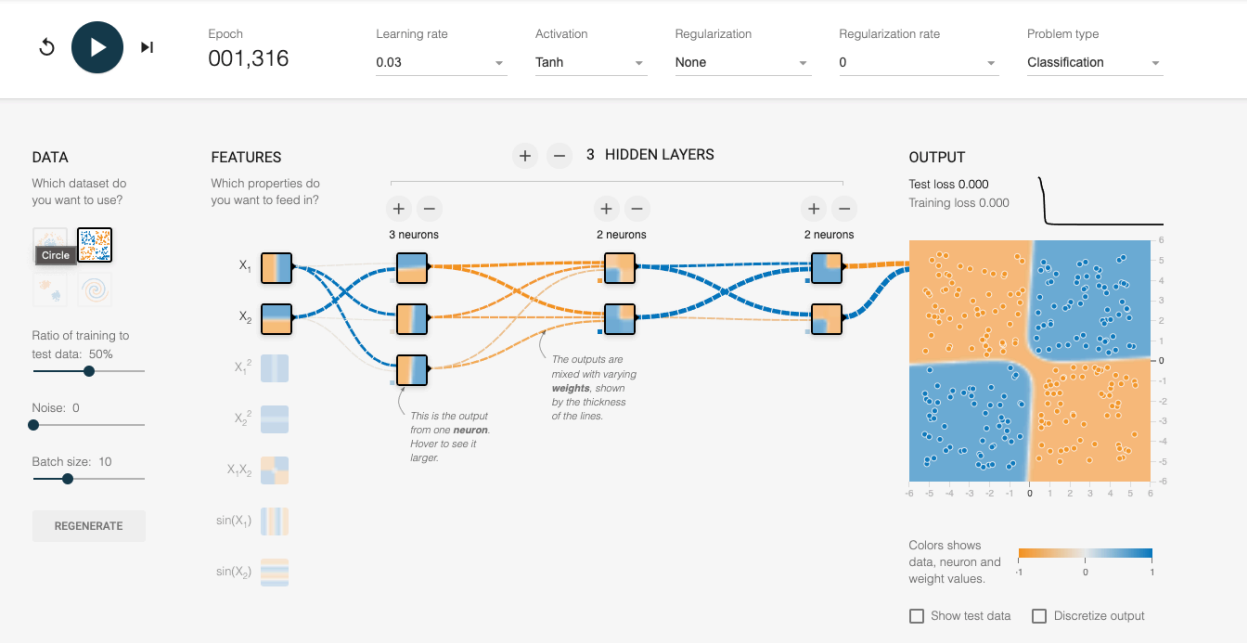
Sin(X2)為 x=sin(y)的值

依我的了解，在測試時會輸入 testing data 不同的特徵進入訓練好的模型，而這些 features 在測試時輸入的部分會用到，至於哪些在測試有用？這會與起初不同的問題有不同的 feature 選擇，進而在測試時只用到那些一開始選擇的 feature。

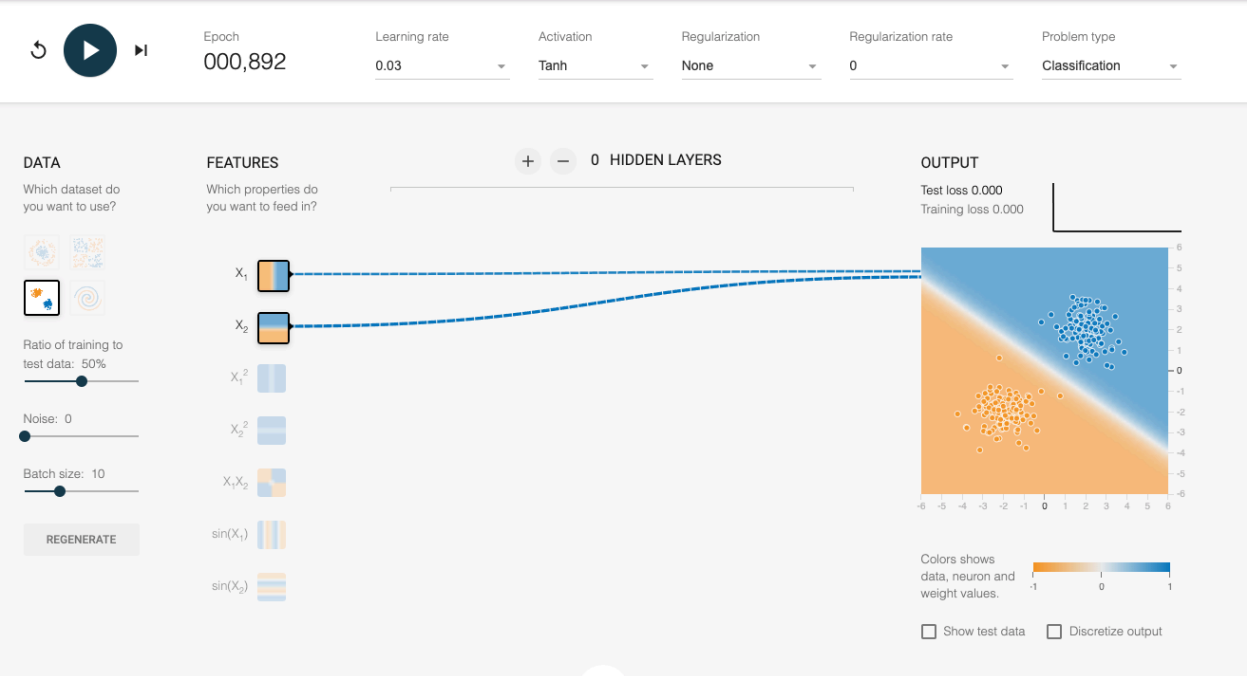
Circle: 運用到 x1, x2，至於為何只用到 x1, x2，我其實不太清楚，因為我都是測試不同 feature 組合去訓練，但我猜測是 circle 此問題沒有很複雜，只需要運用的最基本的 feature 就可以達到不錯的效果，在測試時，可以達到 training loss 與 test loss 都為 0 的結果。



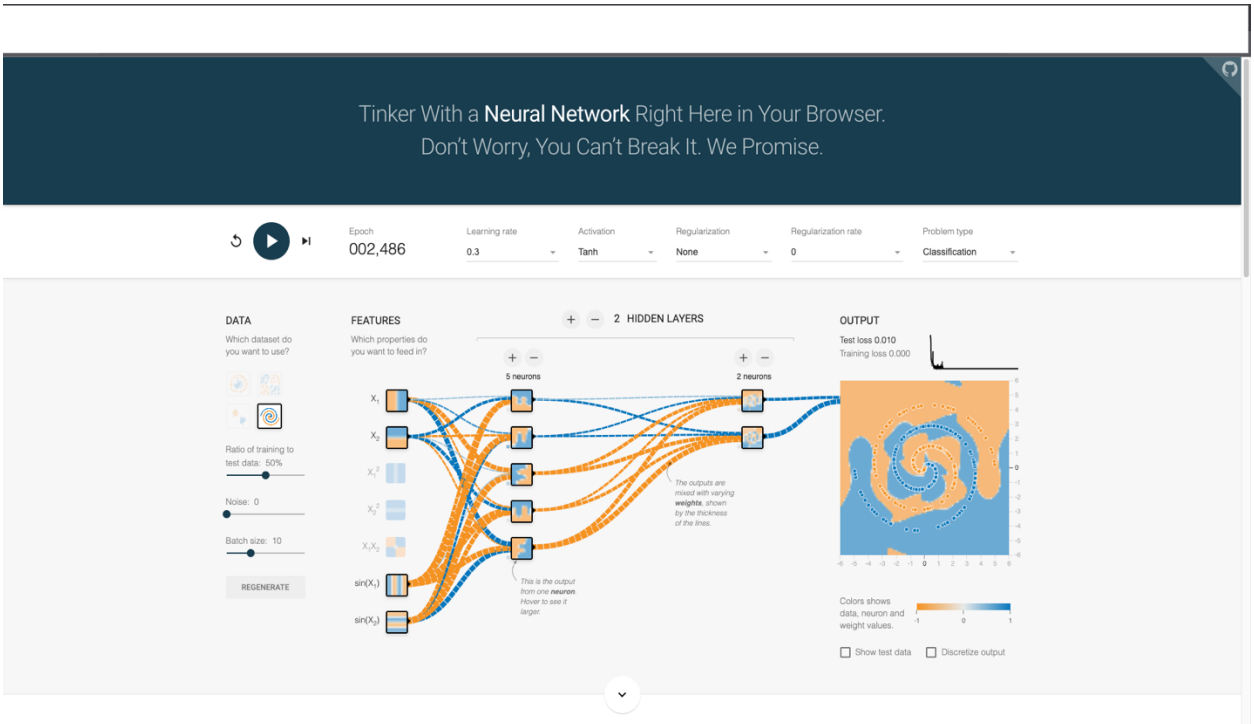
XOR:運用到 x1, x2，在測試時，可以達到 training loss 與 test loss 都為 0 的結果



Gaussian:運用到 x1, x2，在測試時，不需要 layers 與 neurons，就可以達到 training loss 與 test loss 都為 0 的結果



Spiral:運用到 x1, x2, sin(x1), sin(x2)，在測試時，可以達到 training loss 0.01 與 test loss 為 0 的結果



5.

機器學習的任務是找到一個函式，使得輸入經過函式後等於輸出，而輸出的東西會依據問題的不同而有不同 type 的輸出，其中包含 Regression 和 Classification

Regression 是函式輸出一個數值，例如有一個機器學習的任務是預測某個籃球員下一場比賽的得分，假設輸入此球員前一場比賽得分…..等，透過函式會輸出一個數值，例如 12，表示模型預測此球員下一場得 12 分，以上這種類型的任務會歸類為 Regression 任務。

而 Classification 是在有限的選項裡，輸出一個選項，例如有個任務是要判斷輸入的 email 是否為垃圾郵件，而輸出只會有是與否兩種選項，但選項的個數也不侷限只有兩種，像是 alphago 的輸出也是在有限的選項(棋盤的所有位置)中，輸出其中一個選項，以上類型的任務會歸類為 Classification 任務。

6.

(1)Sigmoid

可以將輸入值轉換為 0 與 1 之間的數值，輸入值越大，輸出越接近 1，反之輸入值越小，輸出越接近 0。

用處：由於輸出值得範圍是 0 到 1 之間，處理預測機率適合使用。

缺點：在深層神經網路做 Backpropagation 時，會有梯度消失的問題。

(2)Tanh

可以將輸入值轉換為-1 與 1 之間的數值，輸入值越大，輸出越接近 1，反之輸入值越小，輸出越接近-1

用處：適合使用處理 classification。

缺點：在深層神經網路做 Backpropagation 時，會有梯度消失的問題。

(3)ReLu

此函式輸出為 max(0, 輸入值)，也就是輸入值大於 0 輸出等於輸入，反之輸入值小於等於 0，輸出為 0

用處：目前最廣為使用的

缺點：在 x=0 時，無法微分

(4)Linear

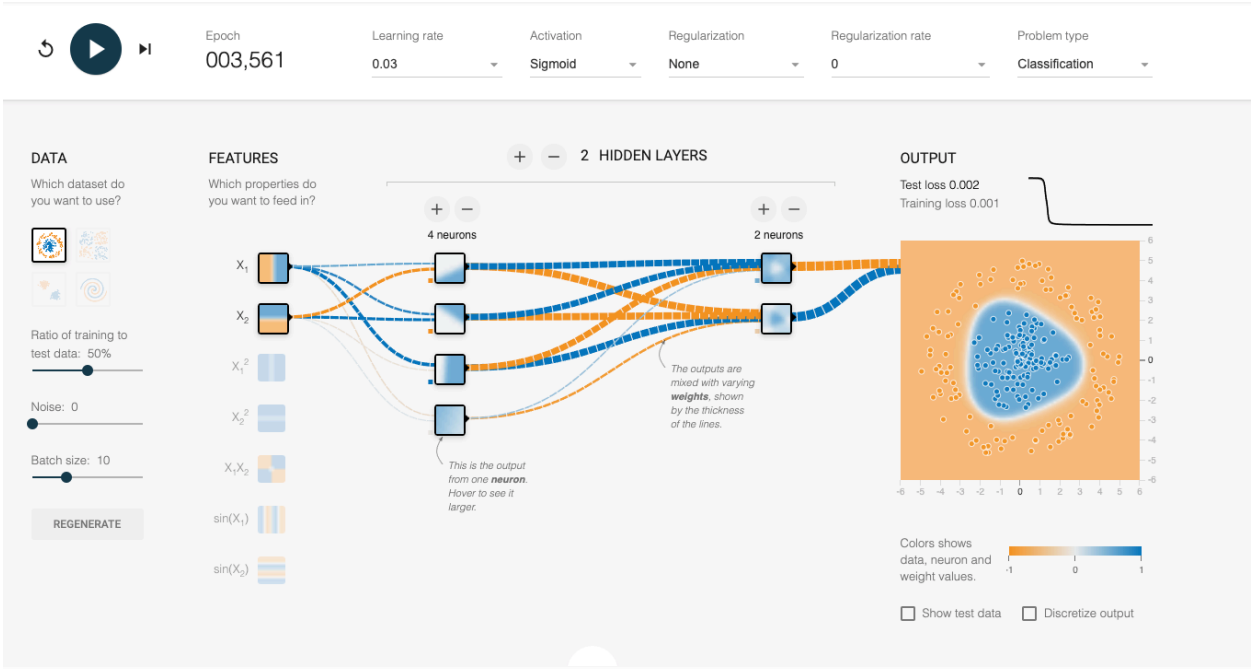
輸出值等於輸入值

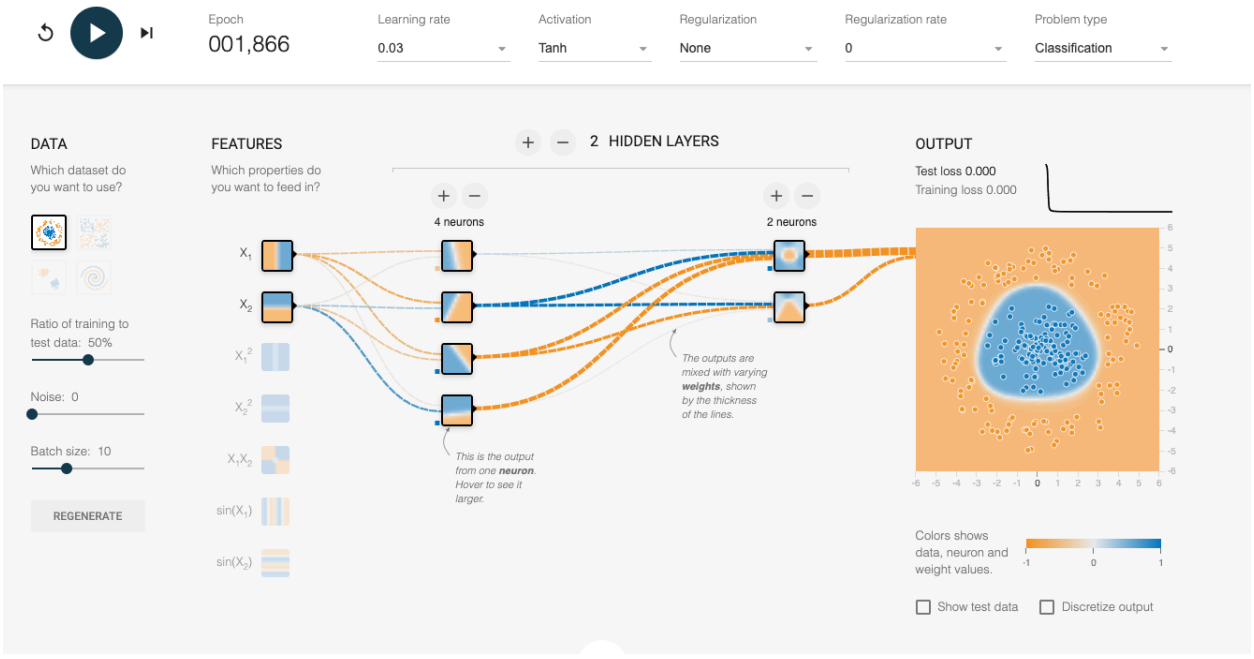
用處：目前較少使用

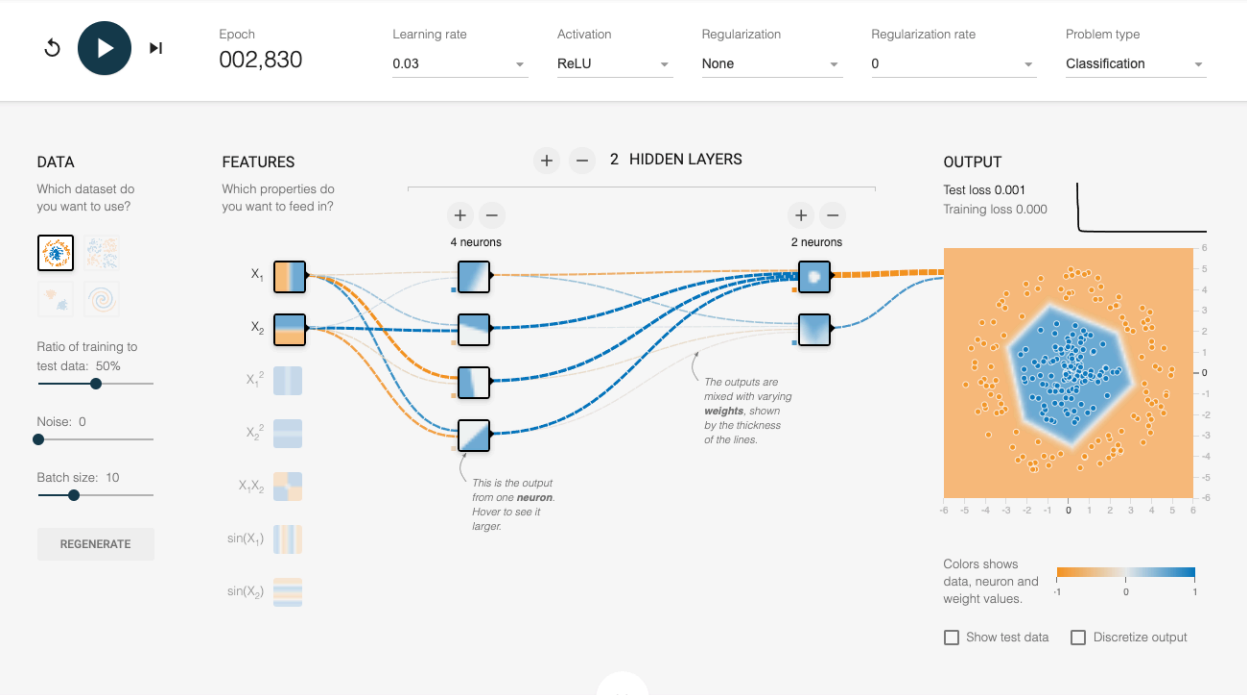
缺點：無法針對非線性關係的問題做學習

以下測試均以第 4 題假設為準，只更改 activation function 去測試。

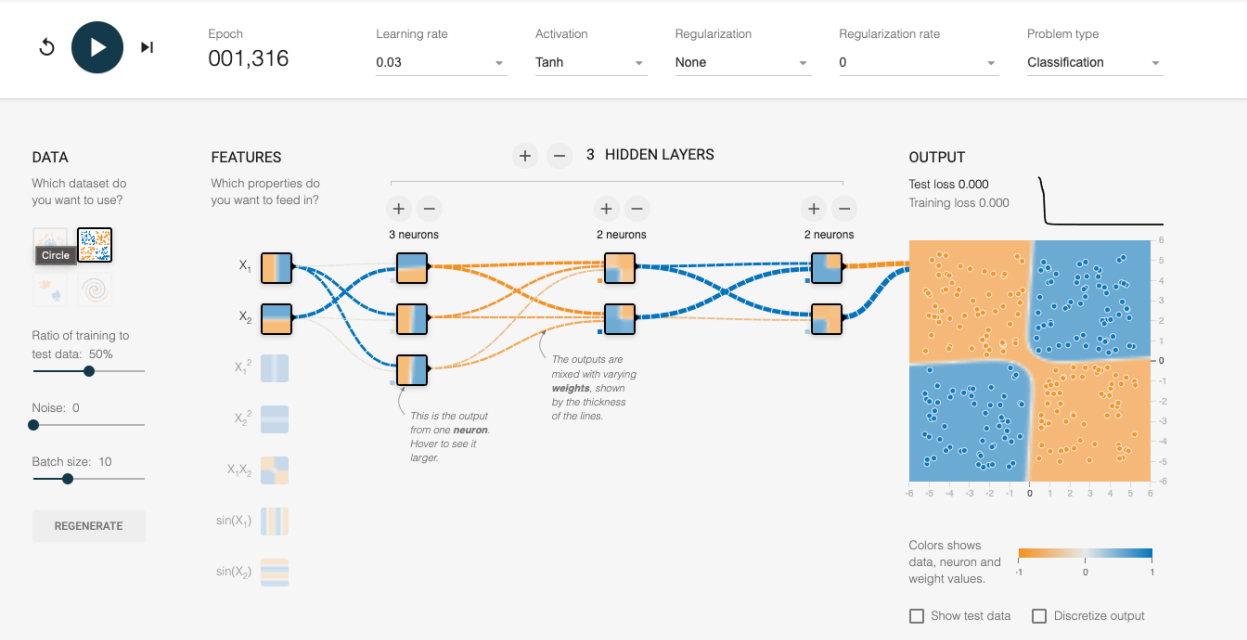
Circle: **適合 Sigmod, Tanh, ReLu，Linear 不適合因為此問題為非線性關係**


Epoch
003,561
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.002
Training loss 0.001
Colors shows data, neuron and weight values.
Show test data
Discretize output

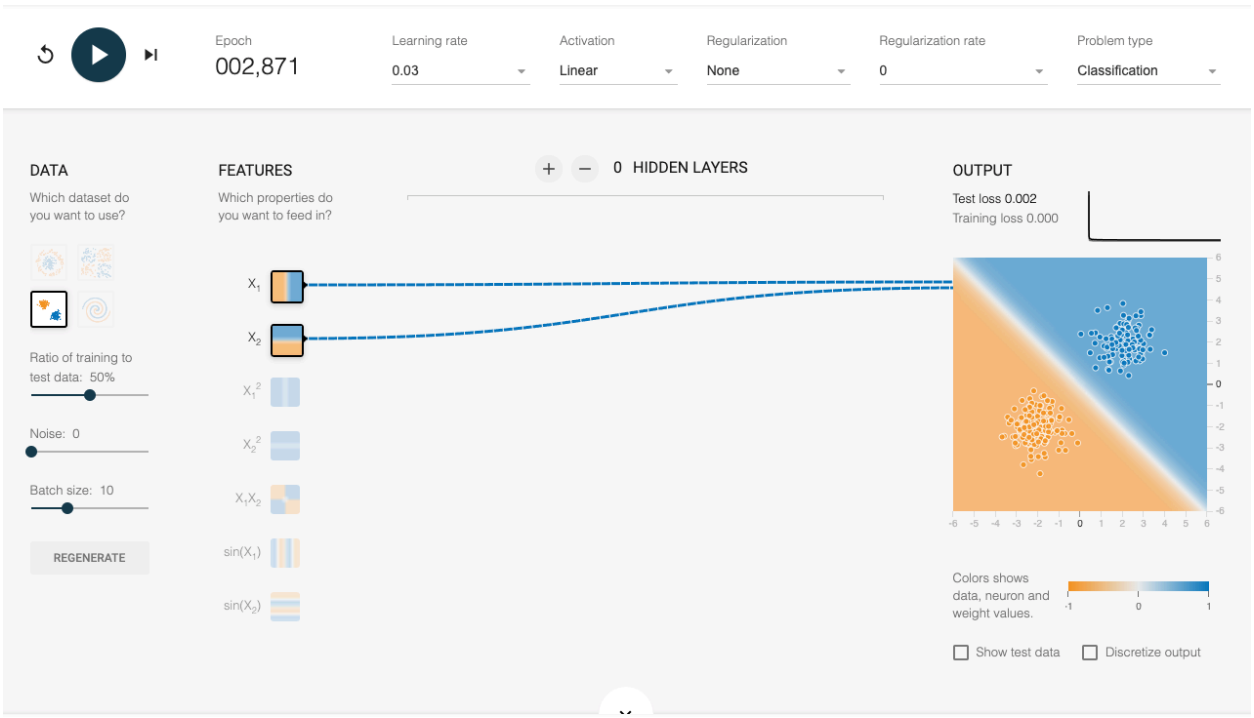

Epoch
001,866
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
The outputs are mixed with varying weights, shown by the thickness of the lines.
This is the output from one neuron. Hover to see it larger.
OUTPUT
Test loss 0.000
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output

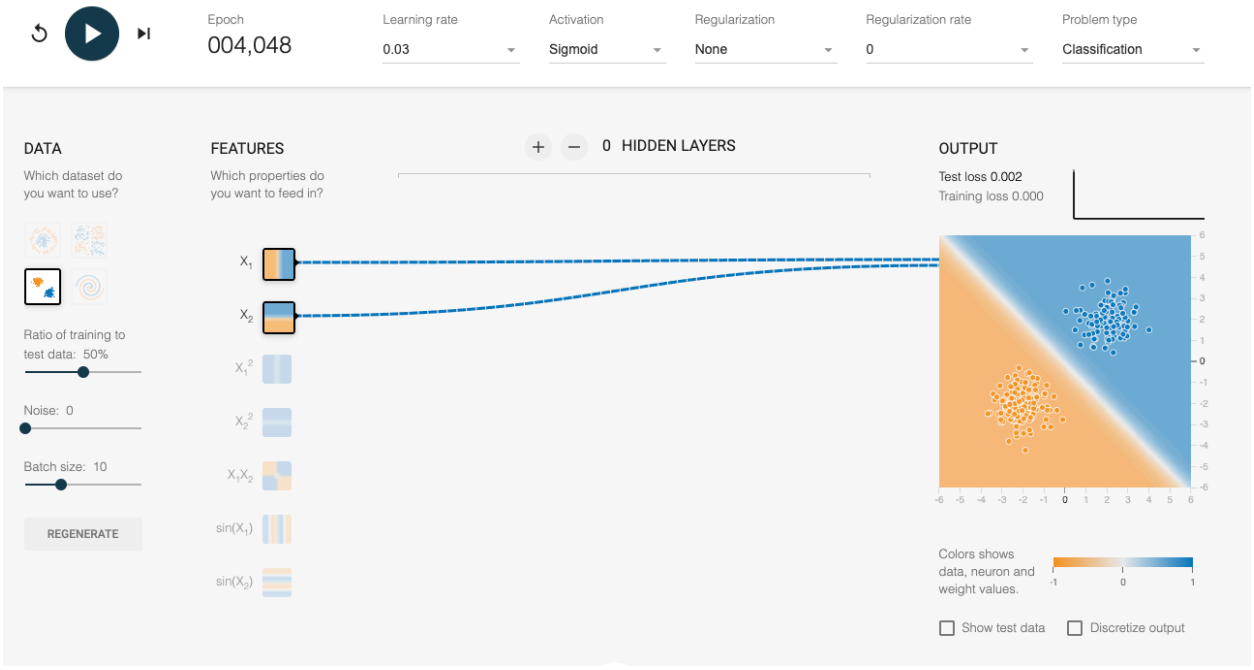


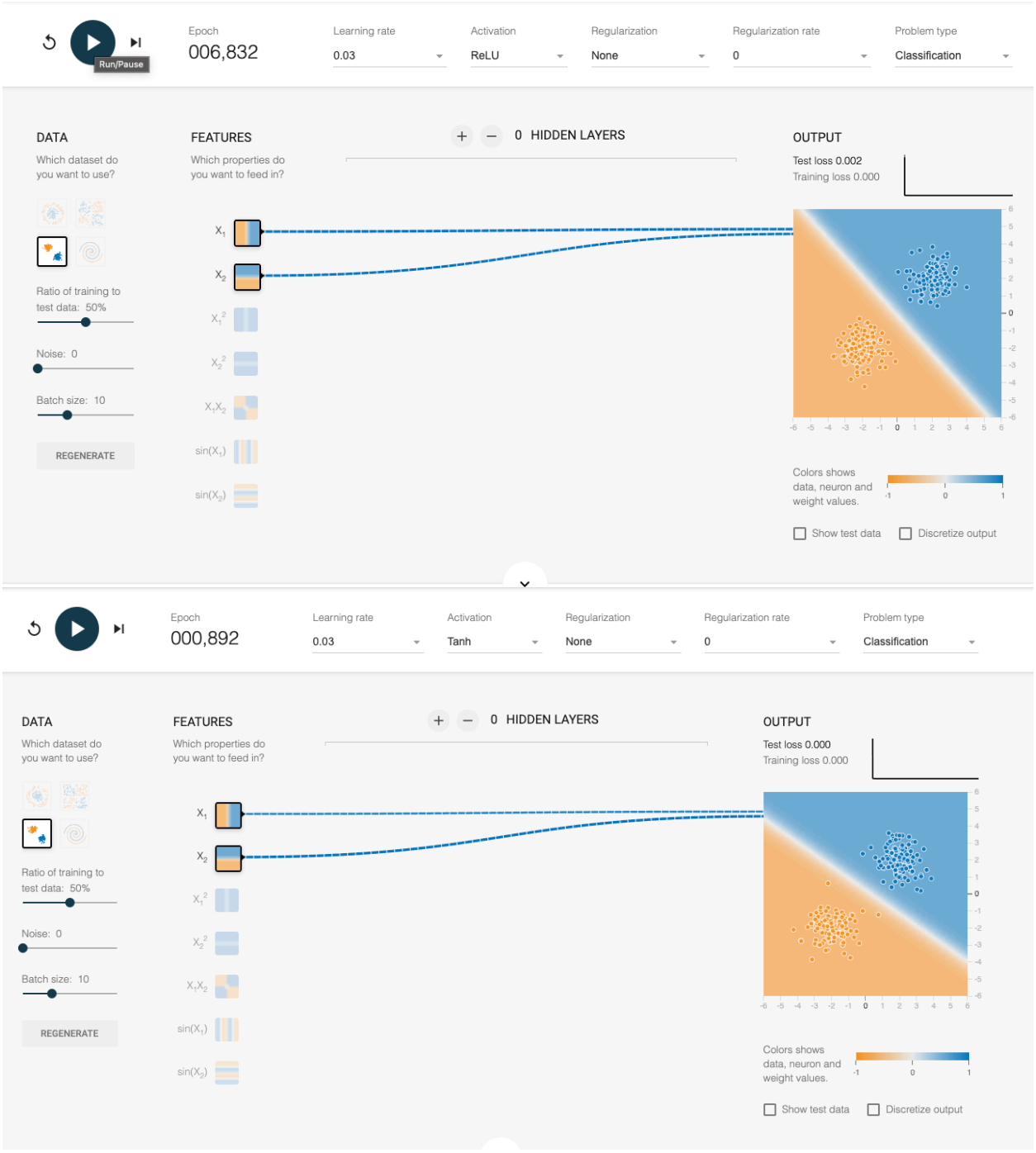
XOR:適合 Tanh，其餘 activation function 測試後的 loss 與 Tanh 的相比有些差距



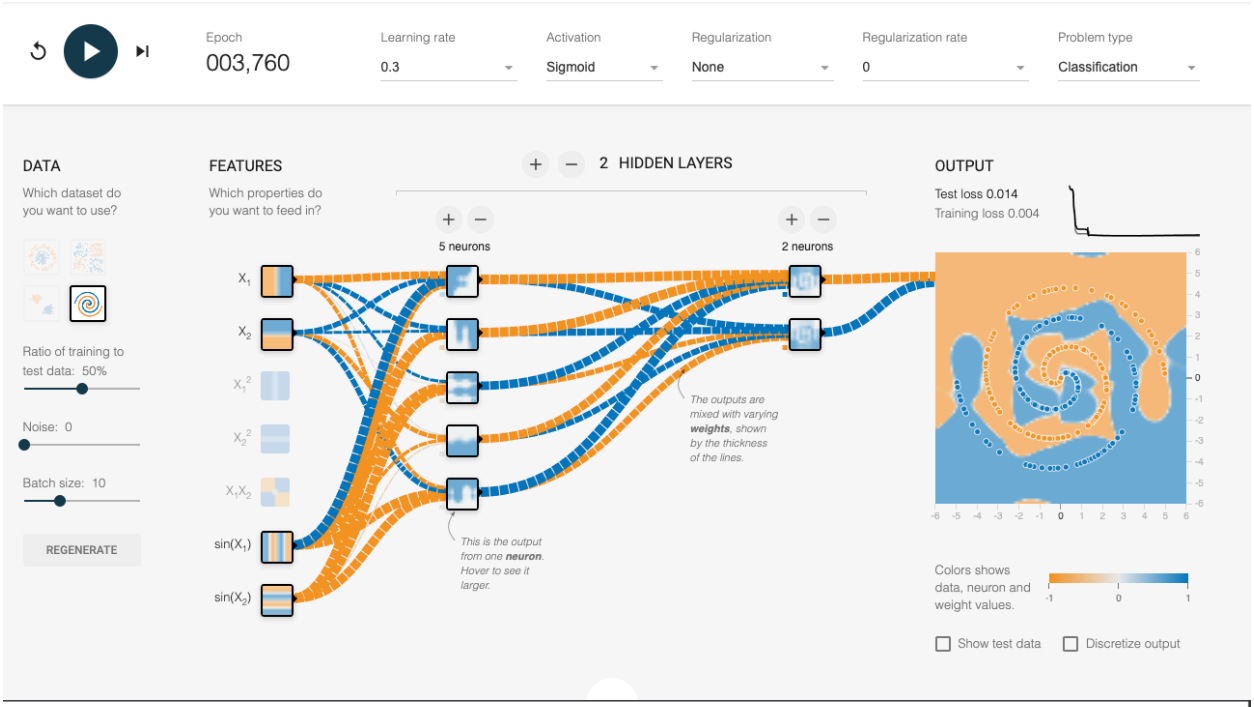
Gaussian: 四種 activation function 都適合

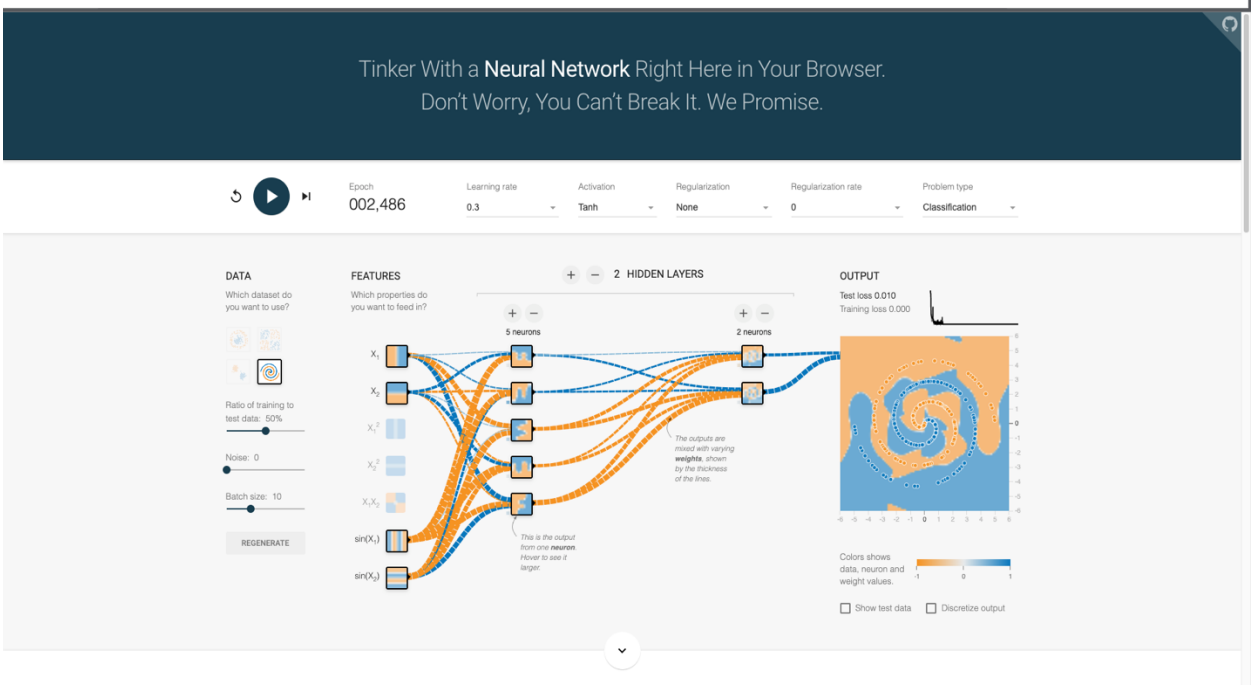

Epoch
002,871
Learning rate
0.03
Activation
Linear
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
OUTPUT
Test loss 0.002
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output


Epoch
004,048
Learning rate
0.03
Activation
Sigmoid
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
OUTPUT
Test loss 0.002
Training loss 0.000
Colors shows data, neuron and weight values.
Show test data
Discretize output



Spiral: 適合 Sigmod, Tanh

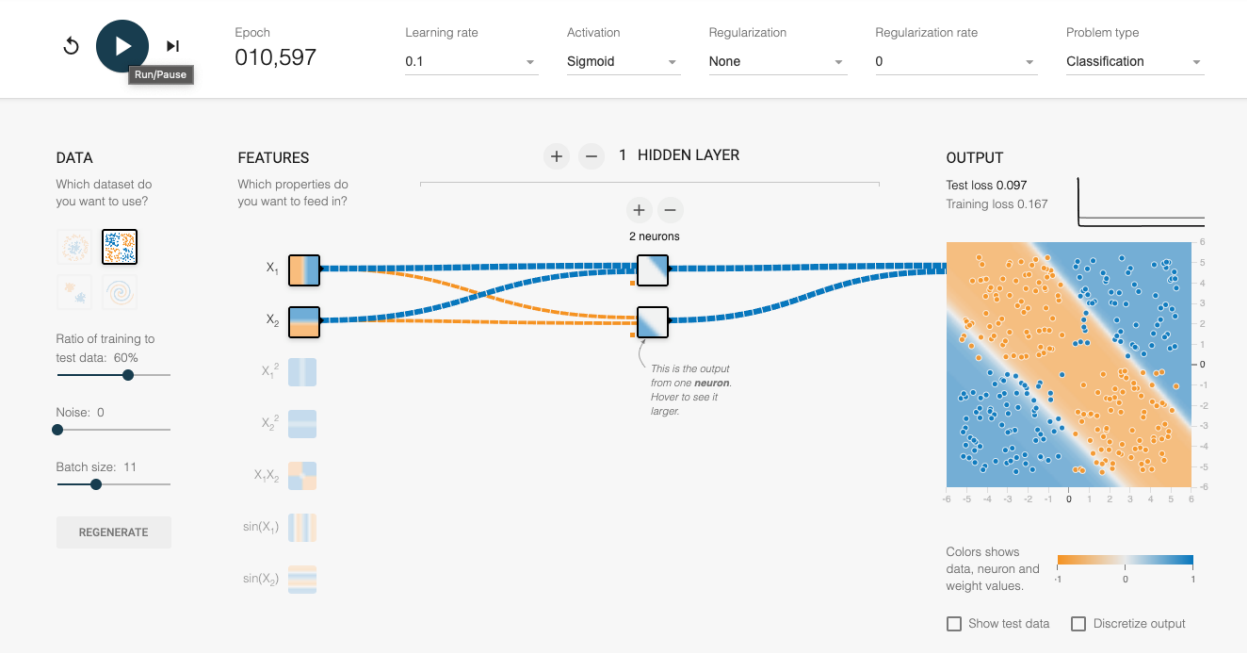




7.

起初在測試時，是直接依照網站預設的值去訓練，再來是更改 training rate 測試一下，發現 trainig rate 為 0.1 時適合此問題，再來是更改權重，權重的選擇參考講義的權重，但是數值是除以 100，因為參考 Tensorflow playground 最後 output 有個隱藏 bias

為 0.1，所以做此決定，另外也調整了 Test data、Noise 和 Batch Size 的數值，觀察訓練後的差異，最後訓練出 Test Loss 為 0.097 與 Training Loss 為 0.167 的模型。



正確率的差異，我都是看 Test Loss 與 Training Loss 的數值，依我的想法，在正常的情況下，兩者數值越低且非常接近，表示此模型的訓練效果越好，而我最後訓練出 Test Loss 為 0.097 與 Training Loss 為 0.167 的模型，發現 Test Loss 比 Training Loss 還小，不確定是否為 underfitting 的情況？

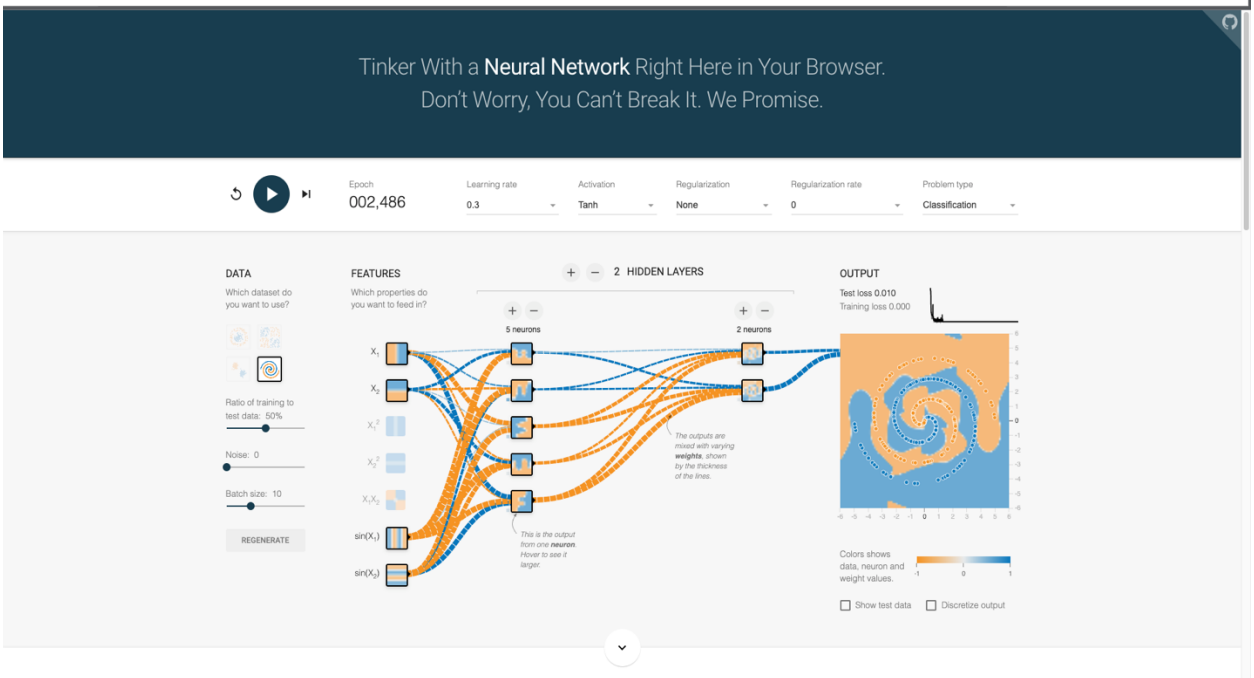
此訓練結果跑了 10597 個 Epoch 的訓練，因為發現誤差圖形沒有降低的趨勢，所以停在 10597。

我認為可以做到誤差在更低的結果，不過需要測試各個參數之間的關係，這也是深度學習最困難的一個項目與目標。

8.

起初對於問題感受較少，所以直接先全選 features，並且用了 3 層 layers，分別各 4 個 neurons，發現雖然 Test Loss 為 0.122，Trainig Loss 為 0.07，但所用的 features 有點多，Layers 與 Neurons 似乎也有減少的空間，在 Feature 部分，我將 X1^2、X2^2 與 X1*X2 不選，前兩個不選的原因是此螺旋狀有負數，不是完全都是正數，而不選 X1*X2 的原因是看螺旋狀的分佈，也不太符合左上與右下是負，右上與左下是正的圖形，而 Layers 改為 2 層，各有 4 個 neurons，最後 Test Loss 為 0.087，Trainig Loss

為 0，有了大幅度的進步，這似乎也說明了在訓練 model 時，不是 features 選越多、layers、neurons 選越多，就會有好的結果。
之後基本上都是 2 層 layers，只在各 layers 的 neurons 數做調整測試，最後在眾多訓練中表現最好的是第一層 5 個 neurons，第二層 2 個 neurons 的 model，Test Loss 為 0.01，Trainig Loss 為 0。



訓練時間運用了 2486 個 epoch，其實在大約 1000 多 epoch 的時候，Loss 的數值就沒再改變了，時間方面不太需要太長。
我認為結果蠻不錯的，如果要選擇在精進的部分，可能是 neurons 的數目有機會再少一點，但此調整可能也會影響到 layer 的數目。

9.

這次訓練與第 8 題的方式比較不同，不是先全選 features，而是直接選擇我認為有用到的 features，一樣不選擇 X1^2、X2^2 與 X1*X2，而 layers 與 neurons 都是 default，2 層 layers，第一層 4 個 neurons，第二層 2 個 neurons，learning rate 與 activation function 也是 default，最後 Test Loss 為 0.013，Trainig Loss 為 0.011，表現蠻出乎預料的，尤其在第一次訓練，之後也只更動 learning rate 與 activation function 下，在

learning rate 為 0.1，activation function 為 Tanh，訓練出 Test Loss 為 0.007，Trainig Loss 為 0.006 的 model。

之後測試都是在以降低 layers 或 neurons 數目下，能否得到此表現或表現更佳？但測試的結果，我目前沒有找到表現更好或有相同表現的 layers 與 neurons 數的組合。

正確率的部分是參照 Training Loss 與 Test Loss 的數值，我自己是沒有計算。

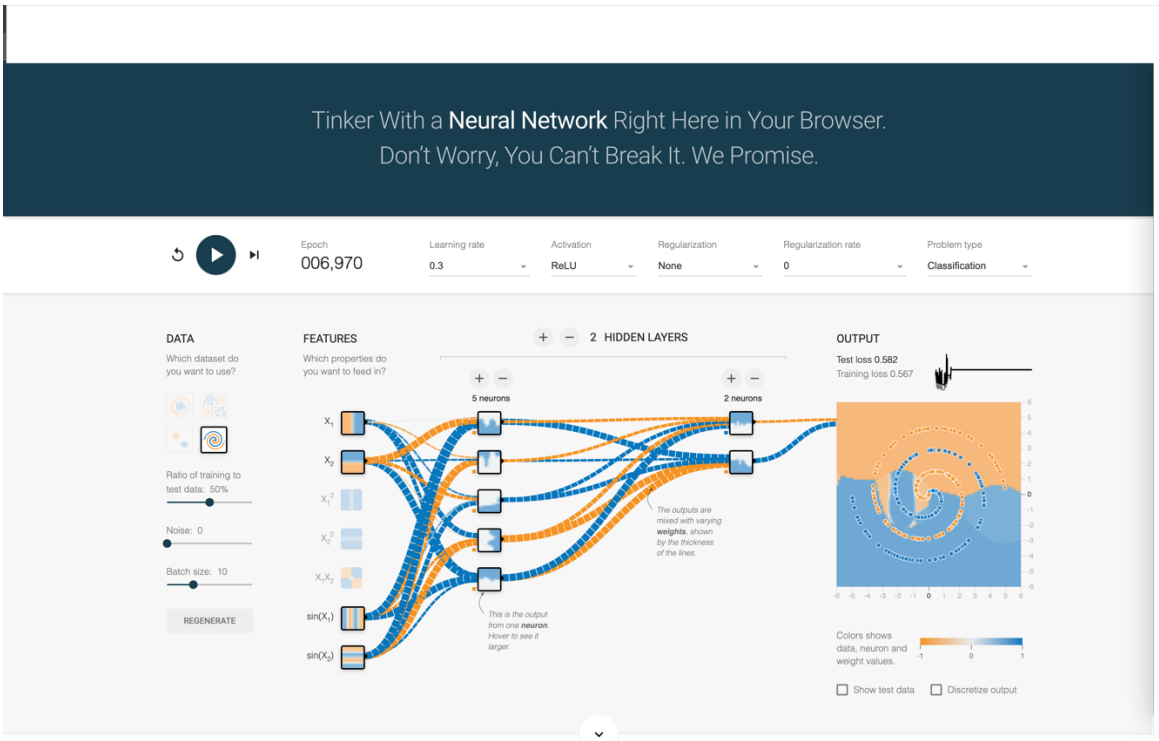
此訓練跑了 3850 個 Epoch，一樣沒有利用到很長的時間。

我認為應該可以再降低誤差。

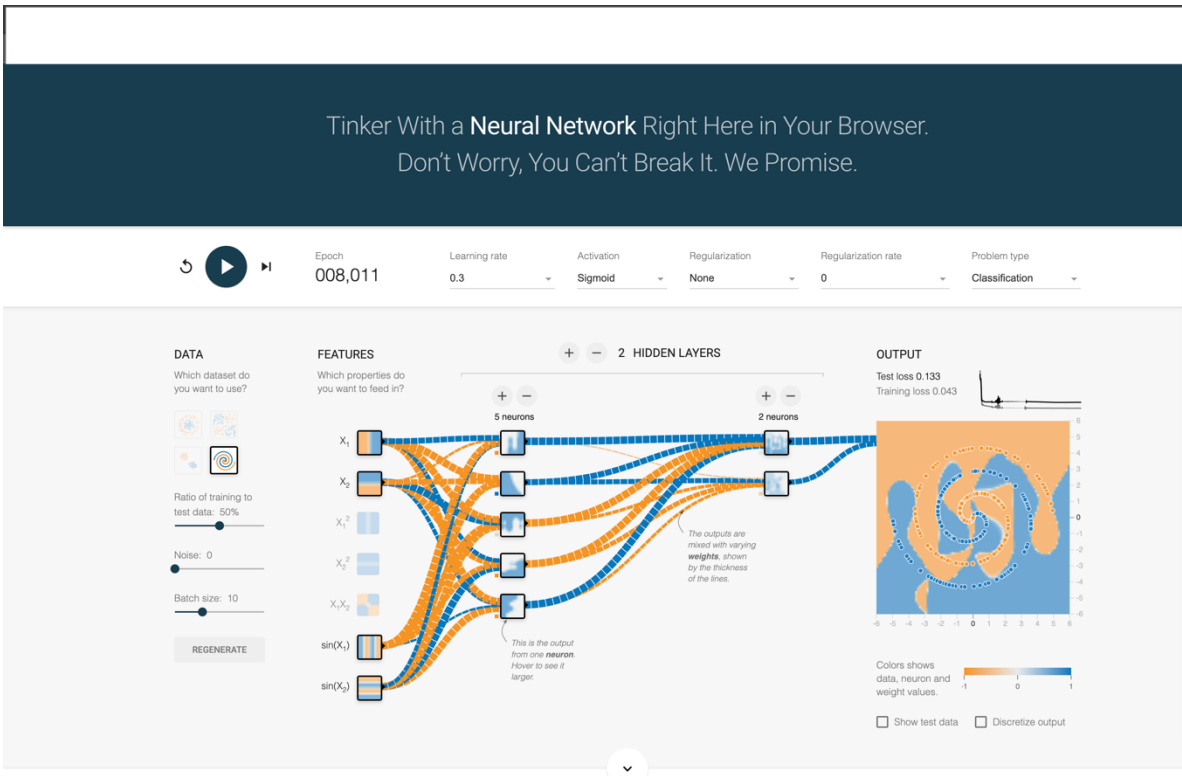
10.

分類問題：在所有設定不變下，只更改 activation function

(1) ReLu：Test Loss 為 0.582，Training Loss 為 0.567，跑了 6970 個 epoch，與原本相比，時間不但變長，Loss 還不減反增

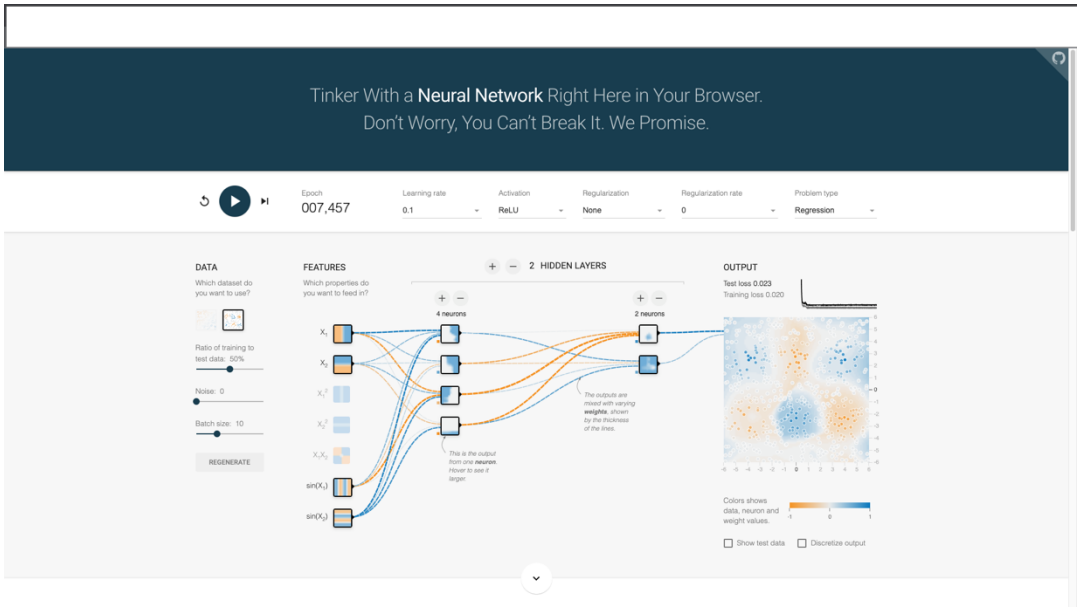


(2) Sigmod：Test Loss 為 0.133，Training Loss 為 0.043，跑了 8011 個 epoch，model 表現雖然沒有原本的出色，但也是蠻低的，訓練時間方面也比原本長



迴歸問題：

(1) ReLu：Test Loss 為 0.023，Training Loss 為 0.02，跑了 7457 個 epoch，與原本相比，時間變長，Loss 雖然增加，但也蠻低的



(2)Sigmod：Test Loss 為 0.009，Training Loss 為 0.007，跑了 6164 個 epoch，與原本的相比，表現不相上下。

11. 最主要的困難是對於之前沒學過的東西，必須自己查資料並試著理解，在過程中時常又出現沒聽過的詞，又得再去查資料，過程中也不確定自己理解的到底是不是正確的，必須利用網路上的資源多方查證。

12.

2. [Day 33] Deep learning -- 訓練技巧 (i) - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)

How to Configure the Learning Rate When Training Deep Learning Neural Networks (machinelearningmastery.com)
Learning Rate in a Neural Network explained - YouTube

3. Fighting Overfitting With L1 or L2 Regularization: Which One Is Better? - neptune.ai
What is Overfitting? | IBM
林軒田機器學習基石筆記 - 第十四講、第十五講 - HackMD

什麼是機器學習中的正則化（Regularization） - 每日頭條 (kknews.cc)

機器學習基石 學習筆記 (4)：機器可以怎麼學得更好? - YC Note (ycc.idv.tw)

機器學習-正規化(Regularization) | MaDi's Blog (dysonma.github.io)

5. 【機器學習2021】預測本頻道觀看人數 (上) - 機器學習基本概念簡介 - YouTube
6. Activation Functions in Neural Networks | by SAGAR SHARMA | Towards Data Science
How to choose best Activation Function for you model | by Siddharth Choudhary | Medium
How to Choose an Activation Function for Deep Learning (machinelearningmastery.com)
7. 使用 TensorFlow 了解 overfitting 與 underfitting | by Airwaves | 手寫筆記 | Medium