Homework 1

Type	Homework
Date	@March 8, 2022

P1

- CPU&GPU: Apple M1 Pro
- RAM: 32GB
- OS: MACOS Monterey

因為需要運行程式碼、修圖、剪輯影片、文書處理等工作，32GB的RAM才能滿足我日常的工作量。

聯絡電話：0978057313

P2

就第6頁的程式來說，我不認為誤差會無限地小下去，因為梯度下降得值在極限處相較來說還是挺大的，且改寫learning rate 後，仍然有可能會錯過誤差最小值。

```
from numpy import *
import pandas as pd
import matplotlib.pyplot as plt


def main():
    losses = pd.DataFrame(columns=["loss"])

    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_weight
    random.seed(1)
    nn_weights = 2 * random.random((3, 1)) - 1
    learningRate = 10

    # train the networks
    for i in range(100000):

        train_out = 1 / (1 + exp(-(dot(train_in, nn_weights))))

        # Run the nn adjustment
        nn_weights += learningRate * dot(train_in.T,  (train_sol-train_out)
                                         * train_out * (1-train_out))

        loss = average((train_sol - train_out)**2 / (2 * train_sol.shape[1]))
        losses = losses.append({"loss": loss}, ignore_index=True)

    losses.plot()
    plt.show()

    test_in = array([1, 0, 0])
    print('\nThe final prediction is ', 1 /
          (1 + exp(-(dot(test_in, nn_weights)))))


if __name__ == "__main__":
    main()
```

P3

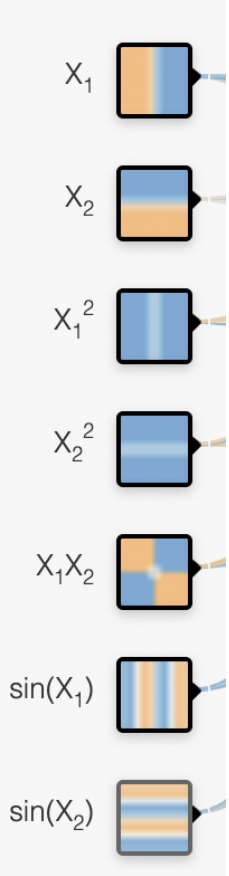
Regularization 是一個讓模型不會 overfitting 的方法，而 L1 & L2 則是 regularization 中的兩種處理方式。而 L1 在做 feature extraction 的時候特別有用，重要的 feature 與不重要 feature 的 weight 會相差較大。

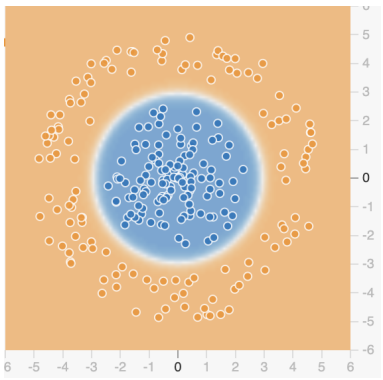
P4

分別為 $x_1, x_2, x_1^2, x_2^2, x_1x_2, sin(x_1), sin(x_2)$

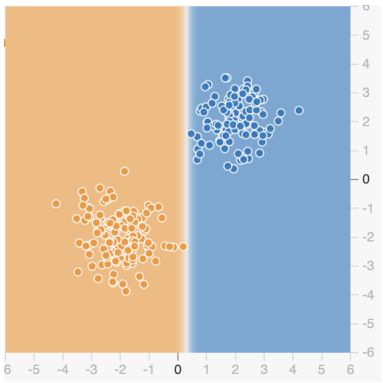
其中藍色代表正數、橘色代表負數。

而我是使用L1 regularization 來判斷說各個 feature 是否用得著。

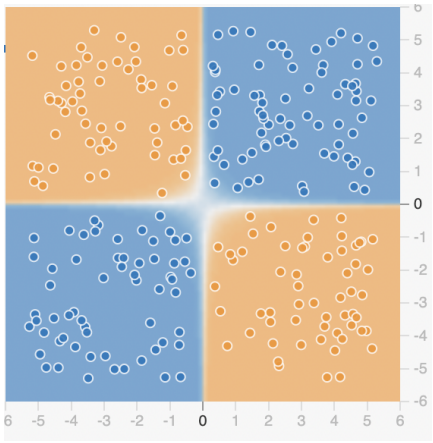




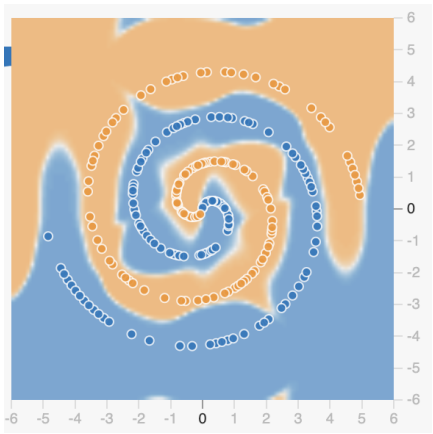
Used feature: x_1^2, x_2^2



Used feature: x_1



Used feature: x_1x_2



Used feature:
$x_1, x_2, sin(x_1), sin(x_2)$

P5

Regression 是在了解兩個變數之間的相關度，不同於 classification 是去判別資料的種類（為離散），regression 更是去預測一個連續的數值。

迴歸分析常納入時間的變數做分析。應用領域如：股票走勢預測、不同時段的交通狀況、原物料價格變化對消費的影響…等。

P6

- ReLU: 會將負數轉換為零
- Sigmoid: 把負數輸出但規範在0~0.5之間，正數輸出在0.5~1之間，有 normalize 的作用。
- tanh: 輸出範圍在-1~1之間，與 sigmoid 很像。
- Linear: ReLu 也是一種linear activation function

經研究發現，4種都可以使用tanh，而圖形較為簡單的則可以使用 linear 的。

P7

可以做出雷同的測試，參考 svitla.com 上面的 how to build neural network on tensorflow for xor。

```
import tensorflow.compat.v1 as tf

tf.disable_eager_execution()

# input X vector
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
# output Y vector
Y = [[0], [1], [1], [0]]

# Placeholders for input and output
x = tf.placeholder(dtype=tf.float32, shape=[4, 2])
y = tf.placeholder(dtype=tf.float32, shape=[4, 1])

# W matrix
W1 = tf.Variable([[1.0, 0.0], [1.0, 0.0]], shape=[2, 2])
W2 = tf.Variable([[0.0], [1.0]], shape=[2, 1])

# Biases
B1 = tf.Variable([0.0, 0.0], shape=[2])
B2 = tf.Variable([0.0], shape=1)

# Hidden layer and outout layer
output = tf.sigmoid(tf.matmul(tf.sigmoid(tf.matmul(x, W1) + B1), W2) + B2)

# error estimation
e = tf.reduce_mean(tf.squared_difference(y, output))
train = tf.train.GradientDescentOptimizer(0.1).minimize(e)

init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for i in range(100001):
    error = sess.run(train, feed_dict={x: X, y: Y})
    if i % 10000 == 0:
        print('\nEpoch: ' + str(i))
        print('\nError: ' + str(sess.run(e, feed_dict={x: X, y: Y})))
        for el in sess.run(output, feed_dict={x: X, y: Y}):
            print('    ', el)
sess.close()

print("Complete")
```

Result

Epoch: 100000

Error: 0.0005582962
[0.02478382]
[0.97827876]
[0.97827876]

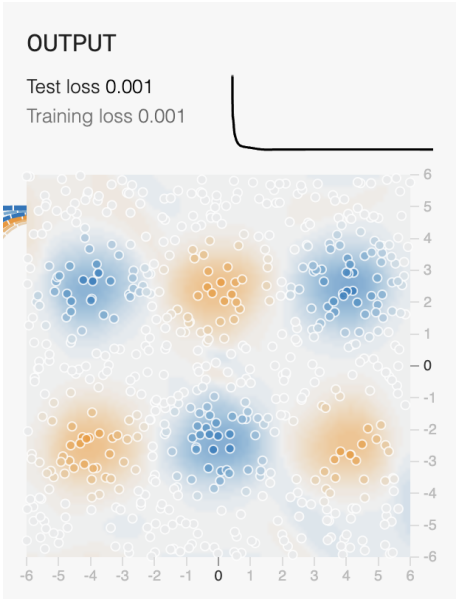
[0.02598696]
Complete

P8

在處理這個分類問題時，我首先加入所有 features 來看哪些 features 較為相關（權重較重），接著選擇較為重要的 features 調整中間的 layers & neurons。過程中有發現越少層 hidden layers 的話，會越穩定的收斂，且在第一層較多layers的話也能提升訓練成功的機率。其中主要判斷好是否成功的依據便是 loss，最低 test loss 可以到 0.002。

P9

同樣先使用L1 regularization 找出相關度較高的 feature($x_1, x_2, sin(x_1), sin(x_2)$)，再增加 layers & neurons 來調整模型精確度，利用 tensorflow playground 的網頁，使用僅僅不到幾秒鐘的時間就完成訓練，也可以達到 test loss 0.001。



P10

原本使用 tanh的話會收斂的比較快，若使用sigmoid 會讓速度較慢（但還是可以收斂），而使用ReLU 收斂時容易不穩定，但結果都還算相似。

P11

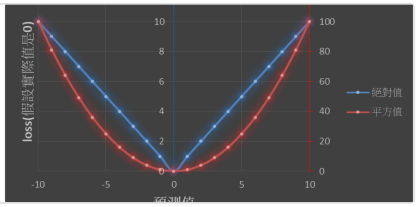
在一開始調整 layers & neurons 的時候，感覺會有點摸不著頭緒，不知道怎麼樣調整才會調整出較好的結果。另外在實作tensorflow 的時候也遇到滿多困難的！需要處理環境、版本等等的問題。

P12 參考網站

機器/深度學習: 基礎介紹-損失函數(loss function)

機器學習大部分的算法都有希望最大化/最小化一個函數/指標，這個函數被稱為「目標函數(Object ...

https://chih-sheng-huang821.medium.com/%E6%A9%9F%E5%99%A8-%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E5%9F%BA%E7%A4%8E%E4%BB%8B%E7%B4%B9-%E6%90%8D%E5%A4%B1%E5%87%BD%E6%95%B8-loss-function-2dcac5ebb6cb

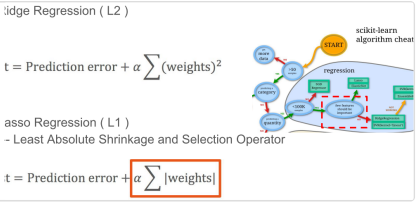


P2 loss function 的設計

Learning Model : L1/L2 regularization差異

一、Weight Regularization (L1/L2)目的 訓練時為了擬合好，會產生很多高次項，但反而容易被雜訊干擾(overfitting)，L1/L2為讓Loss Function更平滑，抗雜訊干擾能力越大。 三、L1/L2產生的高次項係數差異 L1 L2 Lasso(L1) Regularization相較於Ridge(L2)

https://medium.com/ai%E5%8F%8D%E6%96%97%E5%9F%8E/learning-model-l1-l2-regularization%E5%B7%AE%E7%95%B0-8d7fc089b35c



P3 中關於L1 L2 的差異

監督式學習：「分類」和「迴歸」的介紹與比較 - 機器學習兩大學習方法 (一) - iKala Cloud

之前的文章簡介了 AI、機器學習與深度學習。接下來我們會以生活化的情境說明傳統機器學習的方法。本篇首先介紹傳統

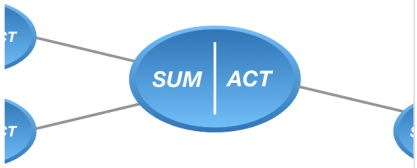
https://ikala.cloud/supervised-learning-classification-regression/

P5 Regression 的定義

【ML09】Activation Function 是什麼?

NN 的 neurons ，除了input node及output node 外，一般除了保存中途的計算結果(SUM)之外還會有Activation Function 的計算(ACT)。它有什麼用？它有什麼形式？如何選擇呢？簡而言之，Activation Function，就是把計算好了的輸入值standardize 好，規範它的「輸

https://medium.com/%E6%B7%B1%E6%80%9D%E5%BF%83%E6%80%9D/ml08-activation-function-%E6%98%AF%E4%BB%80%E9%BA%BC-15ec78fa1ce4



P6 各個 Activation function 的解釋與特性

https://svitla.com/blog/how-to-build-a-neural-network-on-tensorflow-for-xor

P7 中的 xor code