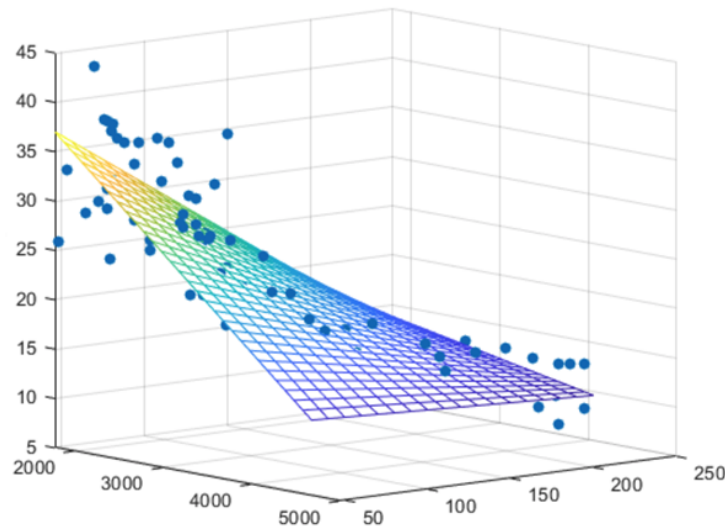


# Assignment 1

This handout is the product of join work by all of the stanCode lecturers



Welcome to SC201!

這份作業將裝備同學們在 [作業二-爛番茄影評AI評價系統](#) 所需的基本技能，包含基礎微積分計算、Python 程式理解、以及 Machine Learning 基本架構與原理

如果作業卡關歡迎各位到社團提問/運用 TA 時間，也非常鼓勵同學們互相討論作業之概念，**但請勿把 code 給任何人看**（也不要將程式碼貼在社團裡）分享您的 code 會剝奪其他學生獨立思考的機會，也會使防抄襲軟體認定有抄襲嫌疑

作業檔案下載

## Problem 1 - validEmailAddress.py

假設我們要判斷一段文字 `str` 是否為 **有效的 email address**，該怎麼請電腦自己判斷呢？

假設我們有一個 10x1 的矩陣（可以把它想成是 10 個 rows、1 個 col 的 list），裡面記錄著 Jerry 認為「可以判斷是否為有效 email address」的特徵與符合每一個特徵可以得到的權重：

Feature Vector	Weight Vector
'@' in the str	0.4
No '.' before '@'	0.4
Some strings before '@'	0.2
Some strings after '@'	0.2
There is '.' after '@'	0.9
There is no white space	-0.65
Ends with '.com'	0.1
Ends with '.edu'	0.1
Ends with '.tw'	0.1
Length > 10	-0.7

在做資料分析，我們常常會將一段資料的特徵萃取出來組合成一個「Feature Vector」，如上圖的左側所示；若將符合特徵所能獲得的權重組合起來，我們則稱之為「Weight Vector」，如上圖的右側所示

我們該如何使用這個 Feature Vector 以及 Weight Vector 來預測一段文字是否為有效的 email address 呢？

假設 `str1 = 'www.google.com'`

我們可以得到 `str1` 的 `feature_vector1 = [0, 0, 0, 0, 0, 1, 1, 0, 0, 1]`

假設 `str2 = 'stancode.tw@gmail.com'`

我們可以得到 `str2` 的 `feature_vector2 = [1, 0, 1, 1, 1, 1, 1, 0, 0, 1]`

當我們把 `feature_vector1` 相對應的特徵（0 代表沒有符合，1 代表有符合）

乘上 `weight_vector` 相對應的特徵權重，我們將得到一個分數 `score1`：

$$0*(0.4) + 0*(0.4) + 0*(0.2) + 0*(0.2) + 0*(0.9) + 1*(-0.65) + 1*(0.1) + 0*(0.1) + 0*(0.1) + 1*(-0.7) == -1.25$$

當我們把 `feature_vector2` 相對應的特徵（0 代表沒有符合，1 代表有符合）

乘上 `weight_vector` 相對應的特徵權重，我們將得到一個分數 `score2`：

$$1*(0.4) + 0*(0.4) + 1*(0.2) + 1*(0.2) + 1*(0.9) + 1*(-0.65) + 1*(0.1) + 0*(0.1) + 0*(0.1) + 1*(-0.7) == +0.45$$

若我們定義 `score > 0` 為有效的 email address；`score <= 0` 為無效的 email address，則 `str1 = 'www.google.com'`，`str2 = 'stancode.tw@gmail.com'` 都可以正確地被我們的程式分辨！這是不是代表：所有的字串現在都可以被電腦正確分辨是否為有效的 email address 了呢 🤖？

請完成 `validEmailAddress.py` 裡的 `def main()`，`read_in_data()`，以及 `def feature_extractor(maybe_email)` 並判斷 Jerry 設定的 `weight vector` 預測準確率是多少？而 Jerry 挑選的 `feature vector` 是否為好的選擇？

您要分析的文字都儲存在 `is_valid_email.txt` 文字檔內。檔案內前 13 筆為 無效的 email addresses；後 13 筆為 有效的 email addresses（如下圖所示）

```

1 "(),:;<>[\]@example.com
2 just"not"right@example.com
3 this\ is"really"not\allowed@example.com
4 email@example..com
5 Abc..123@example.com
6 .stancode@gmail.com
7 stancode.@yahoo.com.tw
8 plainaddress
9 #@%^#$$@#$.com
10 @example.com
11 example.com@
12 www.stanford.edu
13 www.president.gov.tw
14 much."more\ unusual"@example.com
15 very.unusual."@"unusual.com@example.com
16 email@gmail.com
17 firstname.lastname@example.com
18 email@subdomain.yahoo.com
19 firstname+lastname@yahoo.com.tw
20 jerryliao@alumni.stanford.edu
21 liao821226.mse01@g2.nctu.edu.tw
22 liao821226@stanford.edu
23 lifeishard@tsmc.com
24 hotpot@hotmail.com
25 "john..doe"@example.org
26 111@icloud.com

```

validEmailAddress.py 檔案內有**四點請同學們注意**：

(1.) WEIGHT 是一個 10x1 的 list of list (也稱作 2D array / matrix). 這樣建造是有原因的！我們先埋個伏筆，之後再跟同學們解釋

```

WEIGHT = [
    [0.4],
    [0.4],
    [0.2],
    [0.2],
    [0.9],
    [-0.65],
    [0.1],
    [0.1],
    [0.1],
    [-0.7]

```

]

(2.) 在 `def feature_extractor(maybe_email)` 函式裡的程式碼  
`[0] * len(WEIGHT)` (如下圖所示) 是 Python 獨有的語法！可以製造出一個  
長度跟 `WEIGHT` 一樣的 list，而裡面起始值全部都是 0

```
def feature_extractor(maybe_email):
    """
    :param maybe_email: str, the string to be processed
    :return: list, feature vector with value 0's and 1's
    """
    feature_vector = [0] * len(WEIGHT)
    for i in range(len(feature_vector)):
        if i == 0:
            feature_vector[i] = 1 if '@' in maybe_email else 0
        elif i == 1:
            if feature_vector[0]:
                feature_vector[i] = 1 if '.' not in maybe_email.split('@')[0] else 0
            #####
            #                                #
            #                TODO:                #
            #                                #
            #####
    return feature_vector
```

(3.) 在 `i == 0` 的區間內是一個 if else 的超精簡縮寫！大家可以細細玩味這樣的  
寫法跟一般寫法的不同之處 (如下圖所示)

```
def feature_extractor(maybe_email):
    """
    :param maybe_email: str, the string to be processed
    :return: list, feature vector with value 0's and 1's
    """
    feature_vector = [0] * len(WEIGHT)
    for i in range(len(feature_vector)):
        if i == 0:
            feature_vector[i] = 1 if '@' in maybe_email else 0
        elif i == 1:
            if feature_vector[0]:
                feature_vector[i] = 1 if '.' not in maybe_email.split('@')[0] else 0
            #####
            #                                #
            #                TODO:                #
            #                                #
            #####
    return feature_vector
```

(4.) 在 `i == 1` 的區間內有一行 `if feature_vector[0]:` (如下圖所示) 這一行要表達的是：如果在 `feature_vector` 內 0 號 index 的數值為 **1** 就會達到跟 **True** 一樣的效果; 換句話說，如果在 0 號 index 的數值為 **0** 就是 **False**. 在 Python 還有很多東西可以被表達成 `boolean type` ! 像是空的資料結構 ( `[]`, `{ }`, `()`, `...` ) 也是代表 **False**; 而有裝資料的資料結構則可以代表 **True**

```
def feature_extractor(maybe_email):
    """
    :param maybe_email: str, the string to be processed
    :return: list, feature vector with value 0's and 1's
    """
    feature_vector = [0] * len(WEIGHT)
    for i in range(len(feature_vector)):
        if i == 0:
            feature_vector[i] = 1 if '@' in maybe_email else 0
        elif i == 1:
            if feature_vector[0]:
                feature_vector[i] = 1 if '.' not in maybe_email.split('@')[0] else 0
            #####
            #
            #         TODO:
            #
            #
            #####
    return feature_vector
```

最後，請同學計算 Jerry 選擇的 feature vector & weight vector 產出之準確率為多少？請取到小數點下第16位，並將結果填入檔案上方的 `TODO:` (答案若取到小數點下第五位會是0.65384。請特別注意 **There is ‘.’ after ‘@’** 在不只一個 ‘@’ 出現的情況)

```
"""
File: validEmailAddress.py
Name:
-----

This file shows what a feature vector is
and what a weight vector is for valid email
address classifier. You will use a given
weight vector to classify what is the percentage
of correct classification.

Accuracy of this model: TODO:
"""
```

## Problem 2 - validEmailAddress\_2.py

再來要請同學自己設計一個自己的 feature vector & weight vector!

完成了 Problem 1 同學應該會發現準確率不太好，也有一些不合理的權重。請問您該怎麼聰明地選擇，讓 feature vector 與其相對應之 weight vector 產出高於 **85%** 的準確率呢？請將您選擇的 feature vector 在檔案開頭以英文敘述（如下圖所示）

最後再告訴我們：您的 Model 準確率現在是多少？請將答案寫在 **Accuracy of this model:** 後面 (助教會將您設定的模型丟入 26 則 Jerry 重新選的 Emails，測試全班誰的準確率最高！最高準確率的將獲得獎學金 500 元 💰)

```
"""
File: validEmailAddress_2.py
Name:
-----
Please construct your own feature vectors
and try to surpass ~70% accuracy achieved by
Jerry's feature vector in validEmailAddress.py.
feature1:  TODO:
feature2:  TODO:
feature3:  TODO:
feature4:  TODO:
feature5:  TODO:
feature6:  TODO:
feature7:  TODO:
feature8:  TODO:
feature9:  TODO:
feature10: TODO:

Accuracy of this model:  TODO:
"""
```

在 Problem1，為了得到每個文字的 feature vector 與 weight vector 相乘的結果，我們使用從 SC001 第一堂課就學到的 for loop！然而，當數據越來越複雜、資料量越來越大，我們可能就要跟 for loop 說掰掰了👋

取代 for loop 的資料結構我們稱之為矩陣 (array)。這邊幫各位同學複習一下矩陣的基本性質！

在 Python，我們要做一個矩陣，通常會使用一個名為 numpy 的包裹來製造所謂的 **numpy array**

製作一個 numpy array 的方法如下：

```
import numpy as np

weight_vector = np.array([[0.4], [0.4], [0.1], [-0.65], [0.2]])

feature_vector = np.array([[1], [0], [0], [1], [1]])
```

假設我們有兩個 5x1 numpy array (weight\_vector 與 feature\_vector)

**weight\_vector**

0.4
0.4
0.1
-0.65
0.2

**feature\_vector**

1
0
0
1
1



若我們想要將 `weight_vector` 的內容物與其相對應位置的 `feature_vector` 內容物相乘，我們要把放在前面的矩陣變成橫的 (1x5)

把一個 5x1 的 `weight_vector` 轉成 1x5 的矩陣需要使用下列語法：

**`weight_vector = weight_vector.T`**

weight_vector					feature_vector				
0.4	0.4	0.1	-0.65	0.2	1	0	0	1	1

接著，再用下列語法讓兩個矩陣的內容物相乘：

**`score = weight_vector.dot(feature_vector)`**

請同學在這題使用 `numpy` 包裹，練習製作兩個 `array`

(這將是往後課程最重要的核心之一)

另外，同學們若想要讓起始值都是 0 再更改 `numpy array` 的數值，可以使用：

```
import numpy as np

# Initialize an array with 10 rows, 1 col
feature_vector = np.zeros((10, 1))

# Change the value at (row1, col0) from 0 to 1
feature_vector[1][0] = 1
```

## Problem 3 - Machine Learning

這題要請同學仔細閱讀下面的文字：

同學們稍稍回顧一下 Problem1 到 Problem2 經歷的事情

首先，在 Problem1 我們有一個 feature vector 記錄著文字中「哪些細節是跟判斷該文字是否為有效 email address 相關」的矩陣；然後 Jerry 就依自己的直覺猜測了一個 weight vector，記錄著「每一個特徵相對應的權重」（可是瑞凡... Jerry 的直覺不太準...）

然而，假設今天有一位大神跟各位同學說：「我可以讓電腦自己找出一段文字的 feature vector」或是「我可以讓電腦自己找出 feature vector 對應最好的 weight vector」，同學們應該會想：這個人是鬼吧？

沒錯！這些技能就是我們之後要賦予各位同學的，讓每一位同學在八週課程結束後都可以變成自己心目中厲害人才該有的樣子 😊

回到 Problem2。同學們在選擇 feature vector 細節時，應該都是先看「哪些文字被歸類為 True, 而哪些又被歸類為 False」。沒錯，這個過程就是第一堂課 Jerry 提到的 **Supervised Learning** - 電腦也跟各位同學一樣看著提供的答案（我們稱之為 **labels**）來想辦法找到相對應的特徵來符合大多數的例子。一但看出端倪，就馬上修正、跑跑看結果如何！這個也正是電腦在執行機器學習內部發生的事。

最後，當同學在決定 weight vector 的權重時，應該也是看著「哪個 feature 出現一定是 True? 如果有，那我就給它一個 **正整數** 的權重」；或是「哪個 feature 出現一定是 False? 如果有，那我就給它一個 **負整數** 的權重」

其實這段文字要表達的就是往後同學們在學習 AI 心法的一切過程。電腦在學習上是跟人類大同小異的！（這一題就是閱讀 👁️👁️ 沒有 code 要寫～）

## Problem 4 - Derivatives

請將下列4個小題的答案算式過程寫在紙上，並拍照夾帶圖檔到作業資料夾內的 **Problem4** 資料夾 中

若同學想再複習一次微分的基礎概念，下方連結影片是非常好的資源！歡迎同學做練習題之前先觀賞（**4-4.** 會用到上課沒講到的概念，再麻煩參照影片 11:03 開始的部分）

<https://youtu.be/qrBkQ8Ci2fM?t=663>

---

**4-1. Find the derivative of the function with respect to b**

(ANS:  $2\sum(\theta x_i + b)$ )

$$\frac{d \sum_{i=1}^m (\theta x_i + b)^2}{d b}$$

---

**4-2. Find the derivative of the log function with respect to  $h_i$**

(ANS:  $\sum ((-Y_i/H_i) + (1-Y_i)/(1-H_i))$ )

$$\frac{d \sum_{i=1}^m -(y_i \log h_i + (1-y_i) \log(1-h_i))}{d h_i}$$


---

**4-3. Find the derivative of the exponential function with respect to x**  
(ANS:  $(-2X) \cdot e^{(1-X^2)}$ )

$$\frac{d e^{(1-x^2)}}{d x}$$

---

**4-4. Find the derivative of the fractional function with respect to x.**  
(ANS:  $(3X^2-6X+1)/((1-X)^2)$ )

$$\frac{d\left(\frac{1-3x^2}{1-x}\right)}{d x}$$


#### **Problem 5 - linear\_regression\_cubic\_equation.ipynb**

最後一題請同學使用 Anaconda 裡面的 Jupyter Notebook 打開名為 **linear\_regression\_cubic\_equation.ipynb** 的檔案

我們在第二堂課使用了 linear function (一次方程式) 與 quadratic function (二次方程式) 來當預測資料的假設模型 Hypothesis Function。現在，請您試試如果使用 cubic function (三次方程式) 來當作我們預測資料的假設模型是否可以更貼近每一筆資料的趨勢、得到更小的 error/loss 呢？

$$h_{\theta}(x) = \theta_1 X^3 + \theta_2 X^2 + \theta_3 X + b$$

首先，請先完成 `def cost_function`, `def dJ_dtheta1`, `def dJ_dtheta2`, `def dJ_dtheta3`, 以及 `def dJ_db`

```
# This function computes the L2 loss which is
# -----
# [sum of (hypothesis(x_i) - y_i)^2] / 2*m
# where hypothesis(x_i) == theta1(x_i^3) + theta2(x_i^2) + theta1(x_i) + b
# -----
# x_list: list, containing 21 data points
# y_list: list, containing 21 data points
# theta1: float, the parameter of x_i^3
# theta2: float, the parameter of x_i^2
# theta3: float, the parameter of x_i
# b      : float, the parameter that controls the amount of line shift
# return: float, the sum over all L2 losses of data points

def cost_function(theta1, theta2, theta3, b, x, y):
    pass

def dJ_dtheta1(theta1, theta2, theta3, b, x, y):
    pass

def dJ_dtheta2(theta1, theta2, theta3, b, x, y):
    pass

def dJ_dtheta3(theta1, theta2, theta3, b, x, y):
    pass

def dJ_db(theta1, theta2, theta3, b, x, y):
    pass
```

完成後再接續到 `x, y` 數據處理

我們在第二堂課時發現：若我們不對數據平移、縮放，當我們將數據資料平方（甚至像這邊的三次方）資料的數值很容易就變得非常大！導致我們在做 linear regression 的時候要使用很小很小的 step size  $\alpha$  才能控制很大很大的微分項

因此，請對我們這邊的數據點使用 **Normalization** 讓資料數據彼此關係不變，但維持數值在 0-1 之間：

```
# Please do feature scaling by:
# subtracting min(x) from each data point and dividing the result with (max(x)-min(x))
x = # TODO:
y = # TODO:

print(x[10])      # You should see 0.12841038086663803
print(y[10])      # You should see 0.18009827204211928
```

數據處理完成後，請隨機選取一個介於  $-1$  到  $+1$  的數值給我們的四個 parameters: **theta1**, **theta2**, **theta3**, 還有 **b**。當然，為了讓我們 training 過程的 learning curve (Cost-Iteration plot) 可以達到完美的收斂曲線，請選擇一個數值給我們的 hyperparameter  **$\alpha$**  (到底要選 0.1, 0.01, 還是 0.001 呢?)

```
theta1 = # TODO:
theta2 = # TODO:
theta3 = # TODO:
b = # TODO:

# ----- Initialize your alpha ----- #

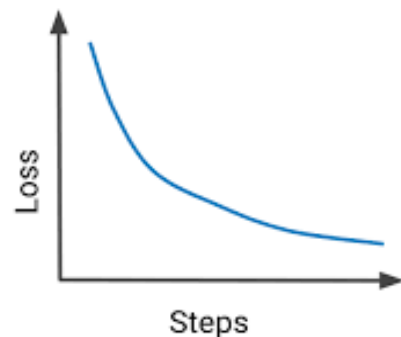
# TODO: which value is appropriate for alpha?
# Try 0.001, 0.01, 0.1

alpha = # TODO:
```

最後請同學完成 Machine Learning 最核心的部分 - Training Process

我們會更新 50 次 **theta1**, **theta2**, **theta3**, 還有 **b** (每一次更新，都會讓曲線更接近我們 Cost 的最小值) 為了方便視覺化，請同學將這 50 次更新的數值全部加入相應的 list 中：**theta1\_history** / **theta2\_history** / **theta3\_history** / **b\_history** / **cost\_history**

若看到檔案最後的 learning curve 如右圖，恭喜！您已經完成作業一了 🎉



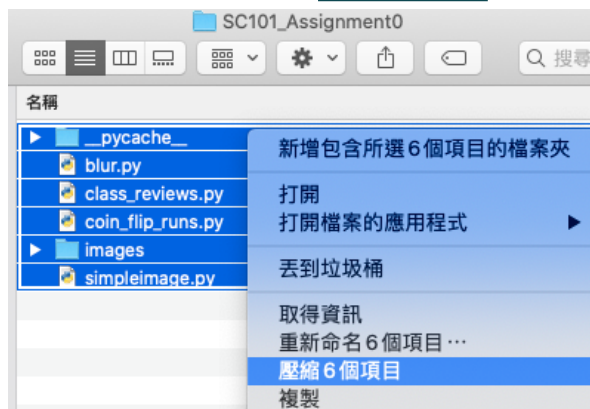
## 作業繳交

恭喜各位同學完成 SC201 Assignment1! 這份作業將帶領同學前往更艱深的觀念 & AI 世界。這份作業的難度與史丹佛大學學生作業非常相似，代表各位同學也成為世界上最強的一群了！

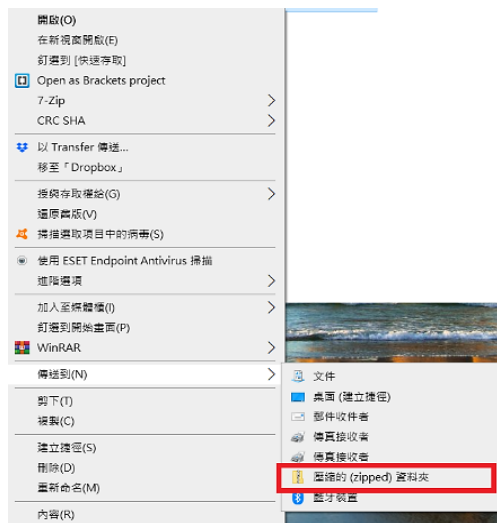
請同學依循下圖，完成作業上傳。

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

macOS：按右鍵選擇「壓縮n個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」



2. 將壓縮檔(.zip)重新命名為「a(n)\_中文姓名」。如：

Assignment 1命名為Assignment1\_中文姓名;

Assignment 2命名為Assignment2\_中文姓名; …

名稱	修改日期	類型	大小
Assignment7_羅志祥	2020/7/1 下午 02:28	檔案資料夾	

3. 將命名好的壓縮檔(.zip)上傳至Google Drive（或任何雲端空間）

1) 搜尋「google drive」

2) 登入後，點選左上角「新增」→「檔案上傳」→選擇作業壓縮檔(.zip)

4. 開啟連結共用設定，並複製下載連結

1) 對檔案按右鍵，點選「共用」

2) 點擊「變更任何知道這個連結的使用者權限」後，權限會變為「可檢視」

3) 點選「複製連結」



5. 待加入課程臉書社團後，將連結上傳至作業貼文提供的「作業提交表單」

stanCode

Should you have any idea or questions, please feel free to contact.