

AI



深度學習

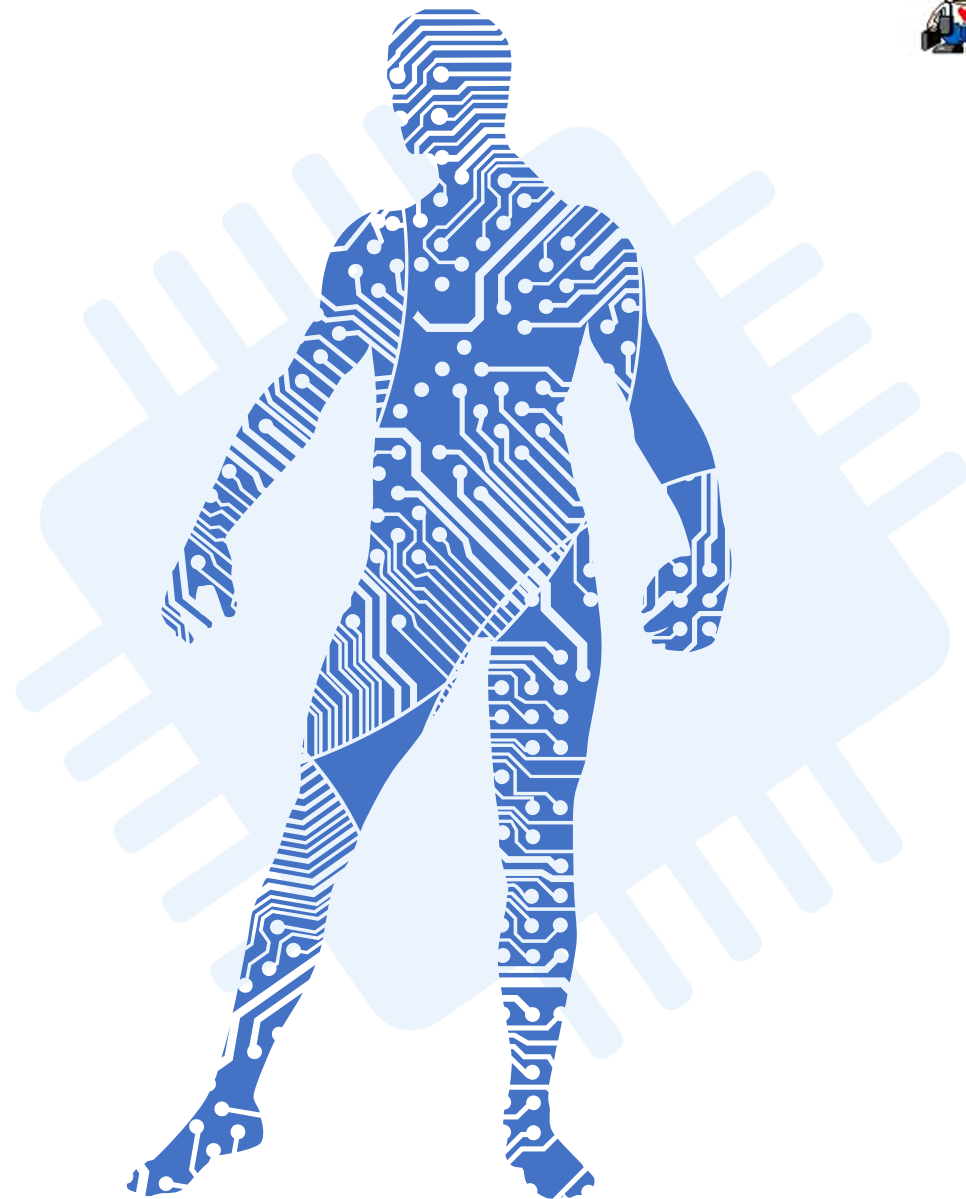
第 2 章 常用函式庫與開發環境

講師：紀俊男



本章大綱

- 神經網路所使用的函式庫
- 雲端開發環境：Colab
- 本章總結





AI

神經網路
所使用的函式庫

有哪些相關函式庫 & 硬體？



高階函式庫



Keras 3.X



Keras 2.x

中高階函式庫



PyTorch



Torch

低階函式庫



TensorFlow



CNTK



MXNet



Theano

硬體層



CPU



GPU



TPU



• TensorFlow

- 開放原始碼的神經網路函式庫
- 2015/11/09 由 Google Brain 計畫發佈
- OS : Win/macOS/Linux/iOS/Android
- 硬體 : CPU/GPU/TPU
- 底層引擎 : C++
- 上層呼叫 : Python, Java, Go, ... etc.
- 最新版 : 2.16 (2024/03)
(2.x 之後整合了 Keras 函式庫)
- 官網 : <https://www.tensorflow.org/>

- 張量 = 高維度數值的集合，方便平行運算用

純量	向量	陣列	張量
Scalar	Vector	Matrix	Tensor
(0-D 張量)	(1-D 張量)	(2-D 張量)	(n-D 張量)
1	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$



- Keras

- 以 Python 撰寫的開源神經網路函式庫
- 2015/03/27，由 Google 工程師弗朗索瓦·肖萊（François Chollet）發佈
- 目的：讓程式師能「無腦」使用神經網路。
- 底層函式庫：
 - TensorFlow、PyTorch、CNTK、MXNet、Theano
 - 不必修改，自由切換。
- 最新版：3.33（2024/04）
（3.x 開始可自由對接 TensorFlow 或 PyTorch）
- 官網：<https://keras.io/>

A 用 Keras 與 TensorFlow 開發差多少？



• 使用 TensorFlow

```
1 import tensorflow as tf
2 import numpy as np
3 import pandas as pd
4 from keras.datasets import mnist
5 (x_train, y_train), (x_test, y_test) = mnist.load_data()
6
7 X = tf.placeholder(dtype=tf.float64)
8 Y = tf.placeholder(dtype=tf.float64)
9 num_hidden=128
10
11 # Build a hidden layer
12 W_hidden = tf.Variable(np.random.randn(784, num_hidden))
13 b_hidden = tf.Variable(np.random.randn(num_hidden))
14 p_hidden = tf.nn.sigmoid( tf.add(tf.matmul(X, W_hidden), b_hidden) )
15
16 # Build another hidden layer
17 W_hidden2 = tf.Variable(np.random.randn(num_hidden, num_hidden))
18 b_hidden2 = tf.Variable(np.random.randn(num_hidden))
19 p_hidden2 = tf.nn.sigmoid( tf.add(tf.matmul(p_hidden, W_hidden2), b_hidden2) )
20
21 # Build the output layer
22 W_output = tf.Variable(np.random.randn(num_hidden, 10))
23 b_output = tf.Variable(np.random.randn(10))
24 p_output = tf.nn.softmax( tf.add(tf.matmul(p_hidden2, W_output), b_output) )
25
26 loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=p_output, labels=Y))
27 accuracy=1-tf.sqrt(loss)
28 minimization_op = tf.train.AdamOptimizer(learning_rate=0.01).minimize(loss)
29
30 feed_dict = {
31     X: x_train.reshape(-1,784),
32     Y: pd.get_dummies(y_train)
33 }
34
35 with tf.Session() as session:
36     session.run(tf.global_variables_initializer())
37
38     for step in range(10000):
39         J_value = session.run(loss, feed_dict)
40         acc = session.run(accuracy, feed_dict)
41         if step % 100 == 0:
42             print("Step:", step, " Loss:", J_value, " Accuracy:", acc)
43
44         session.run(minimization_op, feed_dict)
45     pred00 = session.run([p_output], feed_dict={X: x_test.reshape(-1,784)})
```

46 行

• 使用 Keras

```
1 import tensorflow as tf
2 from tensorflow.keras.layers import Input, Dense
3 from keras.models import Model
4 from keras.datasets import mnist
5 (x_train, y_train), (x_test, y_test) = mnist.load_data()
6
7 l = tf.keras.layers
8
9 model = tf.keras.Sequential([
10     l.Flatten(input_shape=(784,)),
11     l.Dense(128, activation='relu'),
12     l.Dense(128, activation='relu'),
13     l.Dense(10, activation='softmax')
14 ])
15
16 model.compile(loss='categorical_crossentropy',
17               optimizer='adam', metrics = ['accuracy'])
18
19 model.summary()
20
21 model.fit(x_train.reshape(-1,784), pd.get_dummies(y_train),
22         nb_epoch=15, batch_size=128, verbose=1)
```

22 行

長度 TensorFlow : Keras = 2 : 1



- PyTorch

- 由 Meta 開發 (Facebook 母公司) 之開源神經網路框架。
- 程式寫作風格大量使用「物件導向」。常用於高階的神經網路應用，如：影像辨識、自然語言。
- 使用「動態神經網路」架構。與 TensorFlow/Keras 之「靜態神經網路」架構不同，可以在執行時期變更神經網路架構。
- 底層函式庫：
 - Torch
 - Python / C++
- 最新版：2.3.1 (2024/05)
- 官網：<https://pytorch.org/>

- 用 Keras 建構神經網路
- 用 PyTorch 建構神經網路

```
1 model = Sequential()
2 model.add(Dense(100, input_dim=7, activation="relu"))
3 model.add(Dropout(0.5))
4 model.add(Dense(200, activation="relu"))
5 model.add(Dropout(0.5))
6 model.add(Dense(1, activation="sigmoid"))
7
8 # Compiling our model
9 optimizer = SGD(lr = 0.01, momentum = 0.9)
10 model.compile(optimizer = optimizer,
11               loss = 'binary_crossentropy',
12               metrics = ['accuracy'])
```

12 行

```
1 class ANN(nn.Module):
2     def __init__(self):
3         super(ANN, self).__init__()
4         self.features = nn.Sequential(
5             nn.Linear(7, 100),
6             nn.ReLU(inplace=True),
7             nn.Dropout(p=0.5),
8             nn.Linear(100, 200),
9             nn.ReLU(inplace=True),
10            nn.Dropout(p=0.5),
11            nn.Linear(200, 2),
12        )
13     def forward(self, x):
14         x = self.features(x)
15
16         return x
17
18 model = ANN()
```

18 行



- 基於 TensorFlow 的**上層套件**。
- 程式碼好寫，但客製化彈性小。
- 本課程 95 % 的程式都用此函式庫寫成。



- 由 Meta 開發的**神經網路框架**。
- 需有強大的 OOP 背景，常使用於高階的神經網路應用。
- 本課程只會偶而在**影片補充教學**裡用到，不必過度驚慌。



雲端開發環境：
Colab

什麼是「Colab」？



- Google 提供、**免費**的雲端 **Jupyter Notebook** 開發環境



NeuralNet_Mushrooms.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Table of contents

- 神經網路範例（二選一：蘑菇可吃嗎？）
 - 前處理
 - 載入資料集
 - 切分「自變數」與「應變數」
 - 類別資料數位化
 - 降維
 - 切分「訓練集」與「測試集」
 - 特徵縮放
 - 建構神經網路
 - 編譯、訓練、預測
 - 效能評估
 - 結果視覺化
- Section

神經網路範例（二選一：蘑菇可吃嗎？）

這個範例可以用來示範，如何用 Keras 與 TensorFlow 構建神經網路，並做到「機器學習」中「分類」的結果。

前處理

4 13 cells hidden

建構神經網路

```
[ ] from keras.models import Sequential
    from keras.layers import Dense

    # Initialize the whole Neural Networks
    classifier = Sequential()

    # Add the Input & First Hidden Layer
    classifier.add(Dense(input_dim=X_train.shape[1], units=45, kernel_initializer="random_normal", activation="relu"))

    # Add the Second Hidden Layer
    classifier.add(Dense(units=23, kernel_initializer="random_normal", activation="relu"))

    # Add the Output Layer
    classifier.add(Dense(units=1, kernel_initializer="random_normal", activation="sigmoid"))
```

什麼是 Jupyter Notebook ?



- 整合「文字」與「程式碼」於單一頁面

Markdown 語法 : <https://markdown.tw/>

大綱可摺疊

第六章：單純貝氏 (Naive Bayes) 分類器

6-1 單純貝氏分類器假設前提

「單純貝氏分類器」之所以叫「單純」，是因為它假設各個「自變數 X_i 」獨立！正因為自變數各自獨立，所以才冠以「單純」之名。如果自變數有「相依」的情況，那就叫「複雜」、不叫「單純」了！

6-2 使用到的數學定理

「單純貝氏分類器」使用到的數學定理有兩個：「貝氏定理 (Bayes' Theorem)」以及「最大似然率估計 (Maximum Likelihood Estimation, MLE)」。

貝氏定理：

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

最大似然率估計：

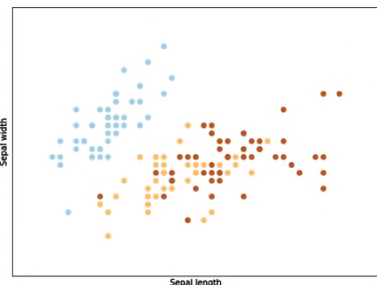
$$\arg \max_{\theta} L(\theta = \hat{\theta}) = \arg \max_{x_i} \prod_{i=1}^k P(x_i)$$

6-3 範例：鸚尾花資料集

接下來我們打算使用「鸚尾花資料集」，來解釋「單純貝氏分類器」。首先，請執行下列這段程式碼，了解一下「鸚尾花資料集」。

```
import matplotlib inline
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.decomposition import PCA

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features.
Y = iris.target
```



6-1 單純貝氏分類器假設前提

「單純貝氏分類器」之所以叫「單純」，是因為它假設各個「自變數 X_i 」獨立！正因為自變數各自獨立，所以才冠以「單純」之名。如果自變數有「相依」的情況，那就叫「複雜」、不叫「單純」了！

6-2 使用到的數學定理

「單純貝氏分類器」使用到的數學定理有兩個：「貝氏定理 (Bayes' Theorem)」以及「最大似然率估計 (Maximum Likelihood Estimation, MLE)」。

貝氏定理：

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

最大似然率估計：

$$\arg \max_{\theta} L(\theta = \hat{\theta}) = \arg \max_{x_i} \prod_{i=1}^k P(x_i)$$

可以用 LaTeX 語法寫公式

LaTeX 產生器 : <https://www.latexlive.com/>

實際看例子 : <https://bit.ly/3JoGIkr>

隨堂練習：了解 Colab 的功能



- 請點擊下列連結，前往講師準備好的 Colab 範例頁面：
 - <https://bit.ly/3JoGlkr>
- 請點擊主標題「Quadratic Equation」旁的三角形圖示，試著摺疊整份文件。
- 請雙擊主標題「Quadratic Equation」，觀察裡面的「Markdown」與「LaTeX」語法。
- 請點擊任何單一儲存格前的「執行」圖示，試著執行一個儲存格看看。
- 請點擊主選單的「執行階段 > 全部執行」，看看是否執行所有程式。
- 您現在了解 Colab (Jupyter Notebook) 能做到哪些事情了嗎？



如何進入到 Colab ?



- <https://colab.research.google.com>

Colab 歡迎畫面 (第一次登入)



Google 帳號



The screenshot shows the Colab 'Welcome To Colaboratory' page. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, a 'Table of contents' sidebar lists: 'Getting started', 'Data science', 'Machine learning', 'More Resources', 'Machine Learning Examples', and a 'Section' button. The main content area is titled 'What is Colaboratory?' and explains that Colab allows writing and executing Python in a browser. It lists three key features: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. Below this, it states that Colab is suitable for students, data scientists, and AI researchers, and provides a link to the 'Introduction to Colab'. A 'Getting started' section follows, describing the Colab notebook as an interactive environment. It includes a code cell with a Python script to calculate the number of seconds in a day, which has been executed, resulting in the value 86400.

CO Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

CO What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

更改檔名

自動存檔

已儲存所有變更

開新檔案

CO PRO

Test.ipynb ☆

檔案 編輯 檢視畫面 插入 執行階段 工具 說明

留言 共用 設定 用戶頭像

RAM 磁碟

在雲端硬碟中尋找
以遊樂場模式開啟

新增筆記本

開啟筆記本 Ctrl+O

上傳筆記本

重新命名

移動

移至垃圾桶

在雲端硬碟中儲存複本
將副本另存為 GitHub Gist
在 GitHub 中儲存副本

儲存 Ctrl+S

儲存及固定修訂版本 Ctrl+M S

修訂版本記錄

我的雲端硬碟 ▾

名稱 ↑	擁有者
影片	我
Classroom	我
Colab Notebooks	我
Google 相簿	我
Google 筆記本存檔	我

檔案所在地



- 請點擊 **File > New Notebook**，開啟一個新的頁面。
- 請將檔名更改為 **Test.ipynb**。
- 請前往自己的 Google Drive 雲端硬碟，開啟下列路徑：
 - 我的雲端硬碟 > Colab Notebooks
- 請觀察一下，是否可以找到 **Test.ipynb**？



分享此頁

CO PRO Test.ipynb ☆

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更

檔案 上傳 刷新 掛載 隱藏

🔍 檔案

{x} .. sample_data

📁

<> 磁碟 199.61 GB 可用

+ 程式碼 + 文字

測試檔

1 print("Hello!")

加上「附註」

留言 共用 設定 用戶頭像

虛擬環境管理

RAM 磁碟

輸入文字即可搜尋指令...

顯示目錄 全域尋找/取代 Ctrl+H 顯示變數檢查器 顯示檔案瀏覽器 顯示程式碼片段窗格 查看資源 Ctrl+Alt+P 上一個儲存格 Ctrl+M P 上傳筆記本 Ctrl+M K 上移所選的儲存格 Ctrl+M N 下一個儲存格 Ctrl+M J 下移所選的儲存格 Ctrl+M I 下載 .ipynb 下載 .py 中斷執行 Ctrl+M I 中斷連線並刪除執行階段

常用代碼 指令面板 終端機面板 (Pro)

[0] da61ba6" 11:42 10-Sep-23

新增文字區塊

+ 程式碼+ 文字

RAM磁碟

刪除文字區塊

↑↓↻💬⚙️✂️📄🗑️⋮

⏏️B*I*<>↻🖼️☰☷☸️⋈Ψ😊☑️

一元二次方程式

使用 Markdown 語法撰寫文字

公式:

$f(x)=3x^2+2x+1$

使用 LaTeX 語法撰寫公式

您的 LaTeX 語法

一元二次方程式

結果預覽區

公式:

$$f(x) = 3x^2 + 2x + 1$$

- Markdown 語法請參考：<https://markdown.tw/>
- LaTeX 語法請參考：<https://is.gd/RPEUIF>

隨堂練習：輸入文字



- 請先保持 **Test.ipynb** 這個檔案的開啟。
- 視情況需要，點擊「**+文字**」，來新增一個文字區塊。
- 輸入下列文字，練習在文字區塊內，使用 **Markdown** 與 **LaTeX** 語法。最終結果應如右側「結果預覽區」所示：

The screenshot shows a Jupyter Notebook interface. At the top, there are tabs for '+ 程式碼' (Code) and '+ 文字' (Text). The current cell is a text cell, indicated by the '文字' tab being active. The cell contains the following text:

```
# 一元二次方程式  
  
公式:  
$$f(x)=3x^2+2x+1$$
```

Below the code editor, there is a preview area showing the rendered output. The title '一元二次方程式' (Quadratic Equation) is displayed. Below it, the text '公式:' (Formula:) is followed by the LaTeX equation
$$f(x) = 3x^2 + 2x + 1$$
.



新增程式區塊

僅執行此區塊程式碼

+ 程式碼

+ 文字

RAM
磁碟

↑ ↓ ↶ ↷ ↵ ↶ ↷ ⋮

一元二次方程式

公式:

$$f(x) = 3x^2 + 2x + 1$$

+ 程式碼

+ 文字

▶

```
1 # 加了這一行, 在 Jupyter Notebook 內繪圖就可以不用 .show()
2 %matplotlib inline
3
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7 x = np.linspace(-1, 1, 50)
8 y = 3*x**2 + 2*x + 1
9
10 plt.plot(x, y)
```

執行所有程式碼

執行階段

全部執行	Ctrl+F9
執行上方的儲存格	Ctrl+F8
執行聚焦的儲存格	Ctrl+Enter
執行選取範圍	Ctrl+Shift+Enter
執行下方的儲存格	Ctrl+F10

輸入 Python 原始碼

開啟 GPU/TPU 加速



• 編輯 > 筆記本設定

編輯

復原插入儲存格	Ctrl+M Z
重做	Ctrl+Shift+Y
<hr/>	
選取所有儲存格	Ctrl+Shift+A
剪下儲存格或選取範圍	
複製儲存格或選取範圍	
貼上	
刪除所選儲存格	Ctrl+M D
<hr/>	
尋找並取代	Ctrl+H
尋找下一個項目	Ctrl+G
尋找上一個項目	Ctrl+Shift+G
<hr/>	
筆記本設定	
<hr/>	
清除所有輸出內容	

筆記本設定

執行階段類型

Python 3

硬體加速器 ?

☐ CPU ☐ A100 GPU ☐ V100 GPU ☒ T4 GPU ☐ TPU

形狀

☒ 大量 RAM

☐ 自動執行第一個儲存格或區段

☐ 儲存這個筆記本時，忽略程式碼儲存格輸出內容

取消 儲存

中斷連線並刪除執行階段

變更執行階段屬性可能會終止目前的工作階段，確定要繼續嗎？

取消

確定

注意：

- 免費用戶不一定搶得到 GPU/TPU。
- GPU/TPU 資源以 Colab Pro 用戶優先。
- Colab Pro 月費 = US\$ 10.49 / 每月

- 請先保持 **Test.ipynb** 這個檔案的開啟。
- 視情況需要，點擊「**+ 程式碼**」，來新增一個程式區塊。
- 輸入下列程式碼，並試著用「區塊執行」與「全部執行」跑跑看：

```
[ ] 1  # 加了這一行，在 Jupyter Notebook 內繪圖就可以不用 .show()
    2  %matplotlib inline
    3
    4  import matplotlib.pyplot as plt
    5  import numpy as np
    6
    7  x = np.linspace(-1, 1, 50)
    8  y = 3*x**2 + 2*x + 1
    9
   10  plt.plot(x, y)
```

- 依照前述投影片的方法，開啟 GPU/TPU，再跑一次看看。



安裝新的外掛套件



```
1 !pip list 列出已安裝的套件
tensorflow 2.13.0
tensorflow-datasets 4.9.2
tensorflow-estimator 2.13.0
tensorflow-gcs-config 2.13.0
tensorflow-hub 0.14.0
tensorflow-tensorflow 2.13.0
tensorflow-tensorflow 2.13.0

1 !pip list | grep keras 檢查有無安裝 Keras 套件
keras 2.13.1

1 !pip install keras 安裝 Keras 套件
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.13.1)

1 import keras
2
3 print(keras.__version__)
2.13.1
```

驗證 Keras 套件安裝成功



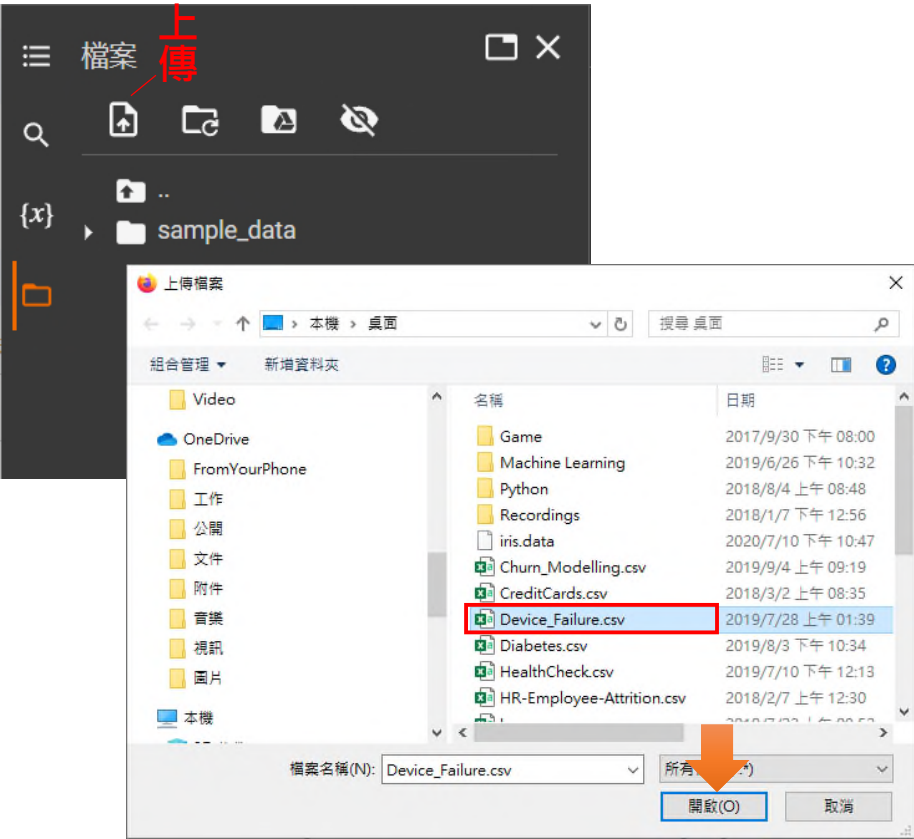
隨堂練習：在 Colab 安裝套件



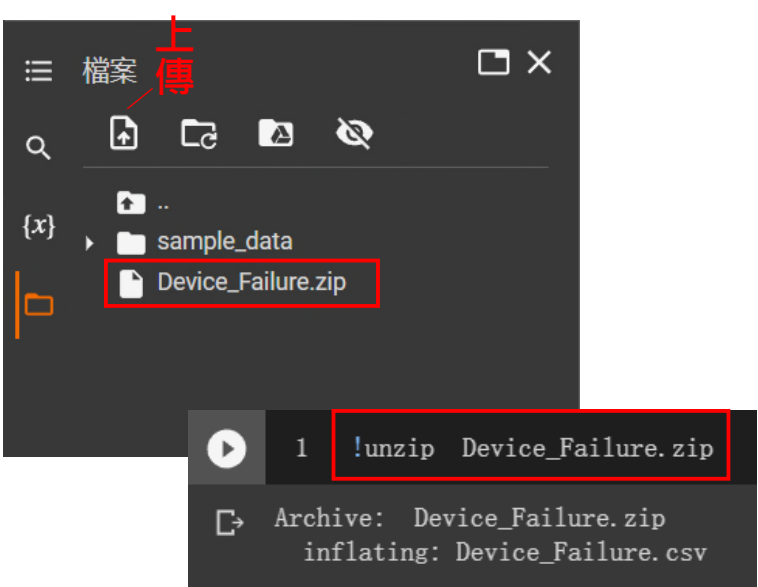
- 請輸入下列指令，列出所有已經安裝的套件：
 - `!pip list`
- 請輸入下列指令，觀看 Keras 套件是否已經安裝：
 - `!pip list | grep keras`
- 請輸入下列指令，安裝 Keras 套件：
 - `!pip install keras`
- 請輸入下列程式碼，以驗證 Keras 已經安裝成功：
 - `import keras`
 - `print(keras.__version__)`
- 事實上，Colab 背後就是一個 Ubuntu 作業系統。任何「不傷害」主機的 Linux 指令，都可以執行！您可以試試看下列指令：
 - `!pwd` (列出「當前工作目錄 Present Working Directory」)
 - `!wget https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data` (可以下載到 iris.data 資料檔)
 - `!git clone https://github.com/wxs/keras-mnist-tutorial.git` (將 GitHub 上的程式碼下載下來)



• 上傳一般檔案

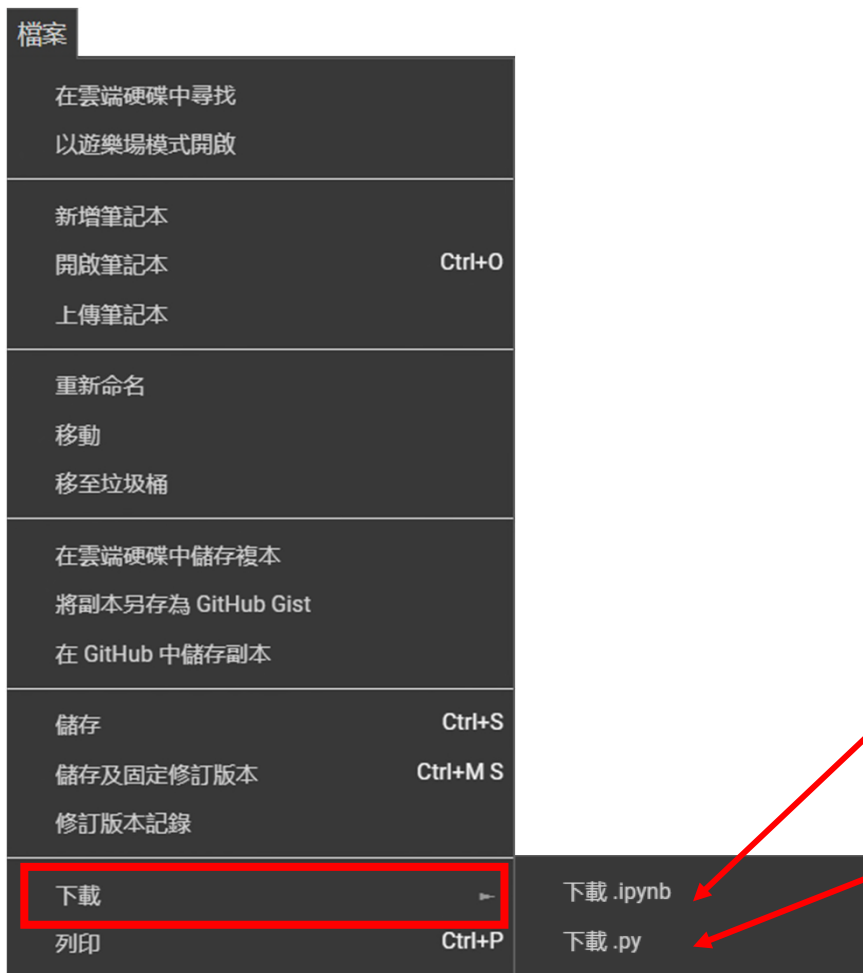


• 上傳壓縮檔案 + 解壓縮



注意：

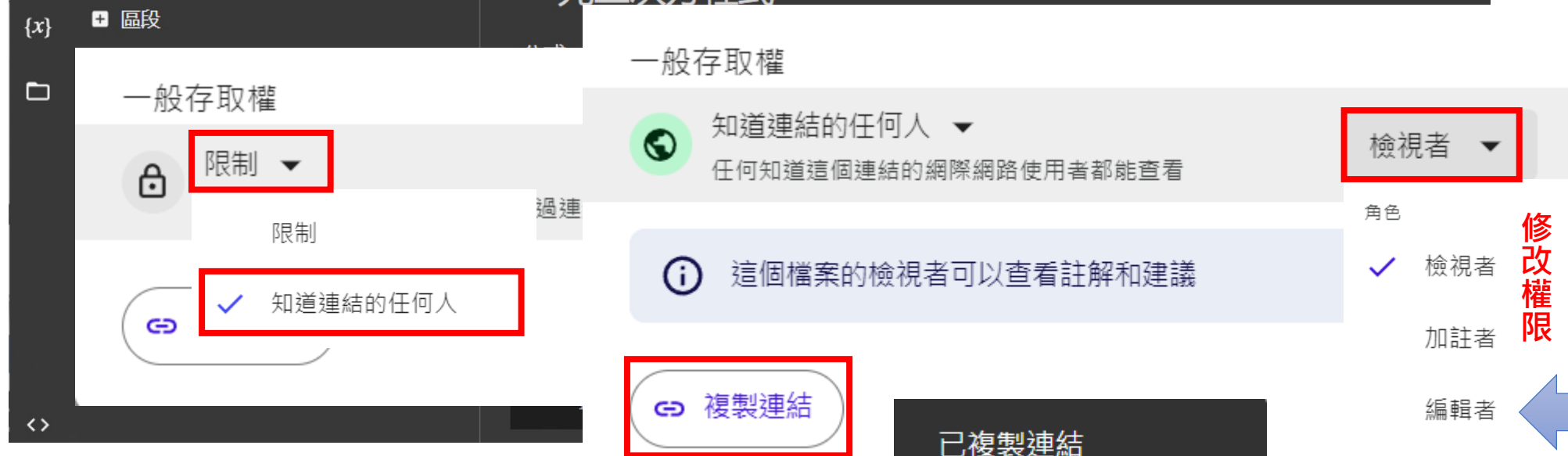
- 目前上傳的檔案，都放在臨時的「虛擬機器」中。只要虛擬機器一卸載（如：30 分鐘沒有動作）就消失。
- 如果要永久保存，建議上載到 Google Drive 再存取。



直接下載成 .ipynb
(可於 Jupyter Notebook 開啟)

轉換成 .py 檔後，再下載
(可於所有 Python 開發環境開啟)

分享 Notebook (繳交作業時可用)



<https://colab.research.google.com/drive/1H1IFcv?usp=sharing>

- 優點

- 免安裝！隨處可用！
- 免費 GPU！不會受限於本地端機器效能，老舊筆電也能跑深度學習程式！
- 可以方便分享成果。

- 缺點

- 需要網路連線。
- 資料集若很龐大（如：照片），上傳很麻煩。
（建議可先上傳至自己的 Google Drive 後，再掛載來避免）
- 掛載的虛擬機器有時限！久沒執行資料直接消失！



- 本課程所使用的函式庫

- Keras、PyTorch。

- 何謂「張量 (Tensor)」

- 高維度數值的集合。

- 什麼是「Colab」？

- 免費的雲端 Jupyter Notebook 開發環境。

- 什麼是 Jupyter Notebook？

- 整合「文字」與「程式碼」於單一頁面的開發環境。

- Colab 基本操作

- 開啟新檔案
- 輸入文字與程式碼
- 開啟 GPU/TPU 加速
- 安裝外掛套件
- 上傳檔案
- 匯出程式碼
- 分享 Colab Notebook 網址

