

# AIoT：樹莓派應用

---

## 補充 A：Python X 機器學習

賴秉樑 debugger

學院創辦人

課程網址 <https://max543.com/debugger>

# Outline

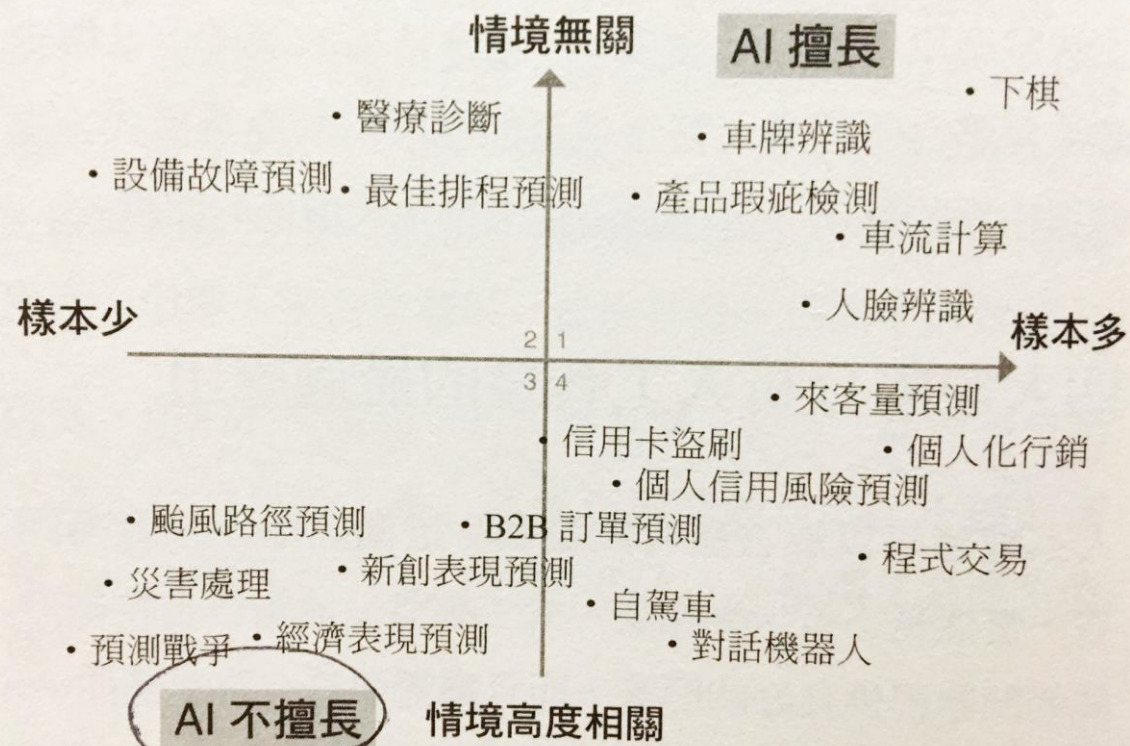
---

- 人工智慧 (Artificial Intelligence, AI)
- 機器學習的原理
- 機器學習的 Hello World!

# 現在的人工智慧

- 現在的 AI 只能針對符合特定形式與條件的問題，提供良好的解答。
- 目前以機器學習為基礎的人工智慧，不可能擁有人類的思考及情緒。

圖 3.1 機器學習擅長解決什麼問題？



# 人工智慧發展簡史

## 第一波

1950-1960

### 符號邏輯

把人的**思考邏輯**放進電腦

由領域專家寫下決策邏輯。

人類還沒辦法清楚理解自己的思考過程，如何告訴電腦？

失敗

## 第二波

1980-1990

### 專家系統

把人的**所有知識**放進電腦

由領域專家寫下經驗規則。

太多難題人類無法解答、無法寫成規則、無法以程式碼表示。

失敗

## 第三波

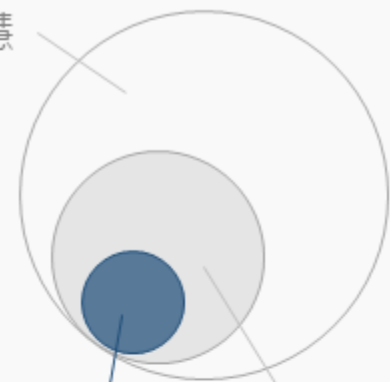
2010-Present

### 機器學習

把人的**所有看見**放進電腦

由領域專家提供歷史資料，讓電腦自己歸納規則。

人工智慧



機器學習  
(第三波人工智慧的代表技術)

深度學習  
(機器學習技術中成長最快、表現最佳)

### 專家系統

專家定義規則

### 傳統機器學習

(與深度學習區隔)

電腦定義規則  
專家定義特徵

### 深度學習

(多層類神經網路)

電腦定義規則 (更準)  
電腦定義特徵

# Outline

---

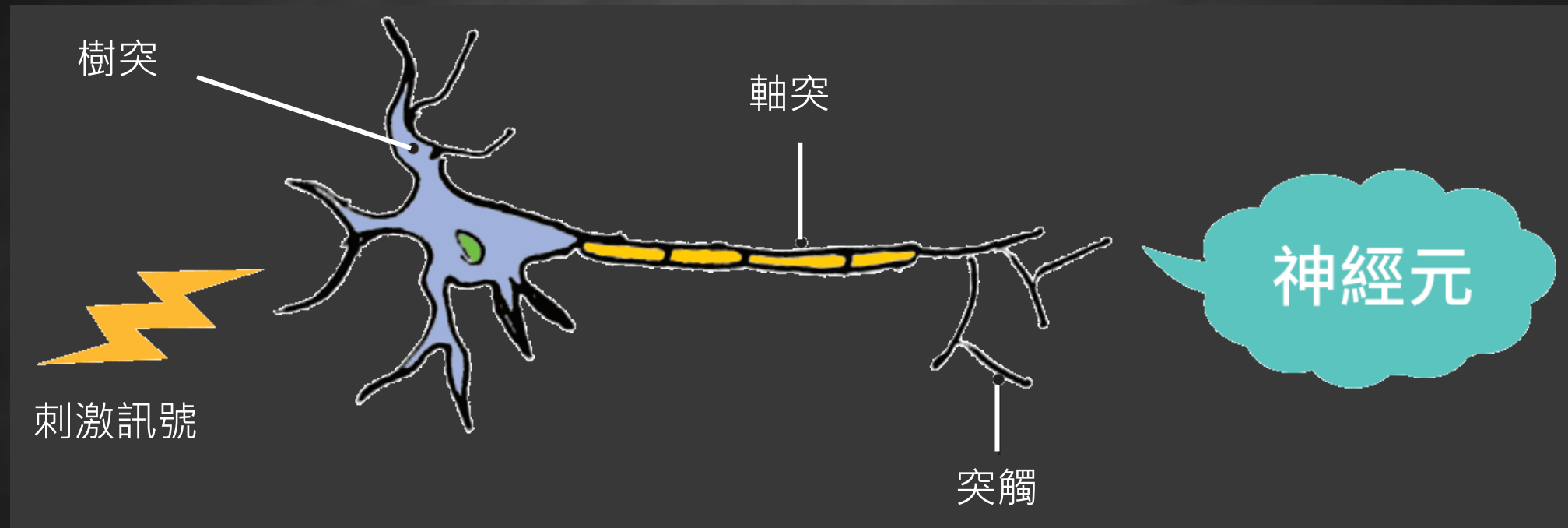
- 人工智慧 (Artificial Intelligence, AI)
- 機器學習的原理
- 機器學習的 Hello World!

# 機器學習技術很成熟

建神經模型，資料就丟進去，電腦自己找解答

# 機器學習的簡單例子

# 初探 AI-神經網路 (主流的機器學習技術)



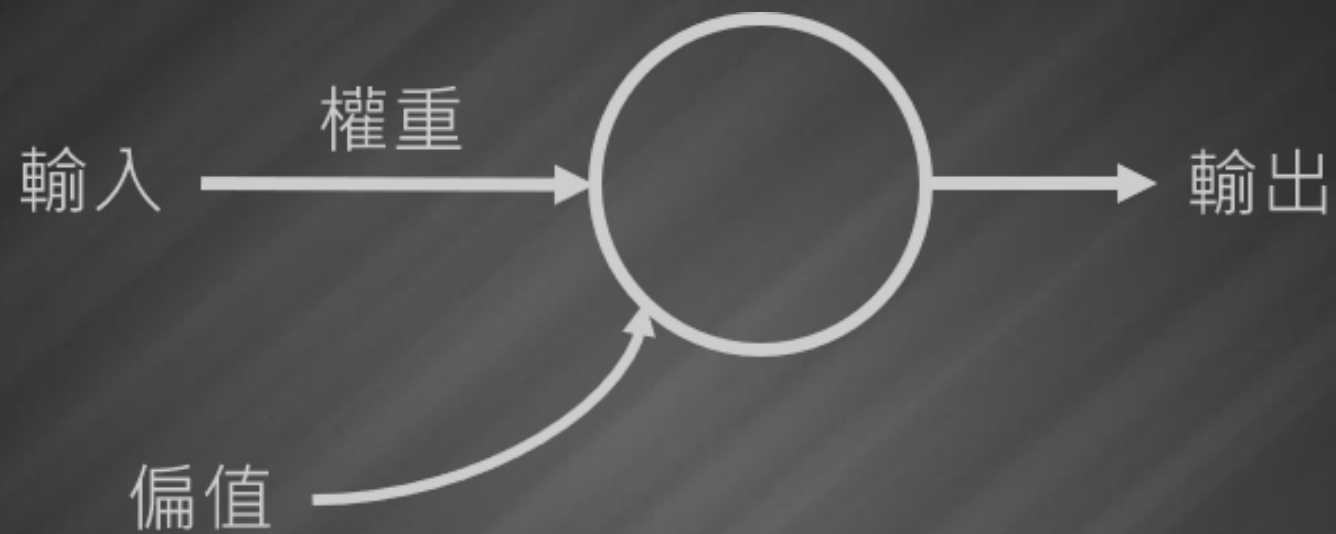
# 人工神經元

1. 輸入：指問題
2. 權重和偏值：自我學習的參數
3. 輸出：解答



$$\text{輸出} = \text{輸入 1} \times \text{權重 1} + \text{輸入 2} \times \text{權重 2} + \text{輸入 3} \times \text{權重 3} + \text{偏值}$$

# 神經元如何學習迴歸問題



$$\text{輸出} = \text{輸入} \times \text{權重} + \text{偏值}$$

# 迴歸問題

$$\text{輸出} = \text{輸入} \times \text{權重} + \text{偏值}$$

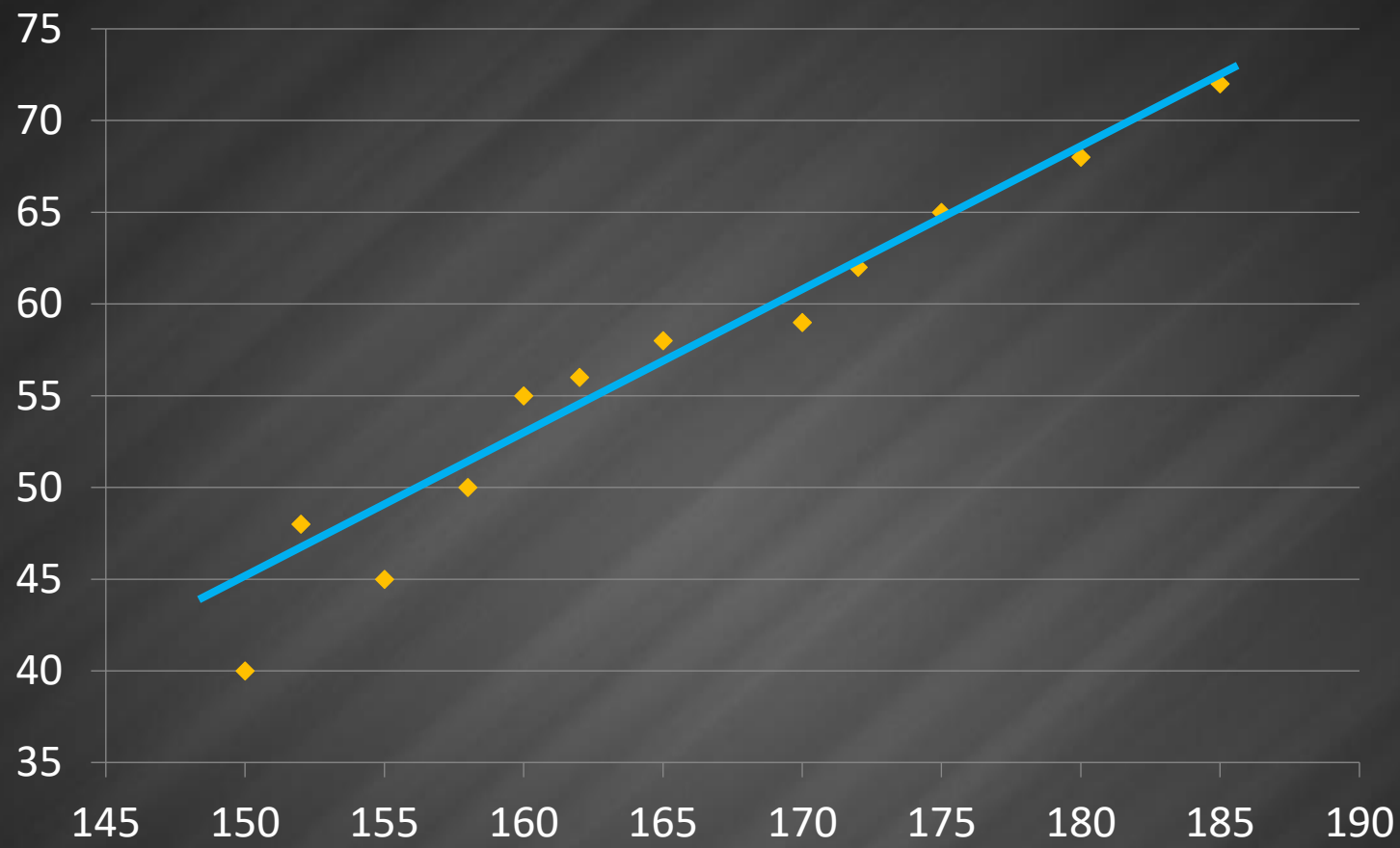
某一班學生的身高和體重是否相關?

能否用身高來推測某位學生的體重?

| 身高  | 體重 |
|-----|----|
| 150 | 40 |
| 152 | 48 |
| 155 | 45 |
| 158 | 50 |
| 160 | 55 |
| 162 | 56 |
| 165 | 58 |
| 170 | 59 |
| 172 | 62 |
| 175 | 65 |
| 180 | 68 |
| 185 | 72 |

# 迴歸線 (函數)

$$\text{輸出} = \text{輸入} \times \text{權重} + \text{偏值}$$



$$y = 0.8337x - 81.331$$

# 迴歸線 (函數)

$$\text{輸出} = \text{輸入} \times \text{權重} + \text{偏值}$$



$$\text{體重} = \text{身高} \times 0.8337 - 81.331$$

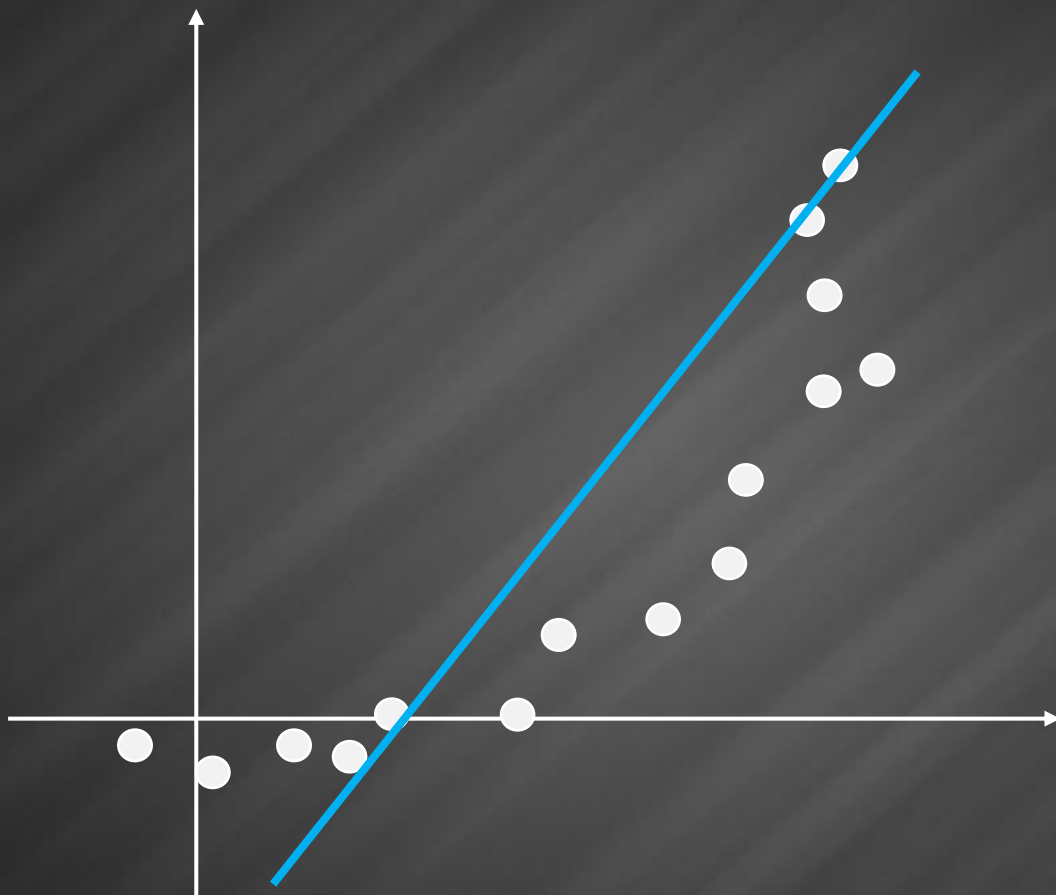
建立兩組資料間的對應函數

# 非線性問題

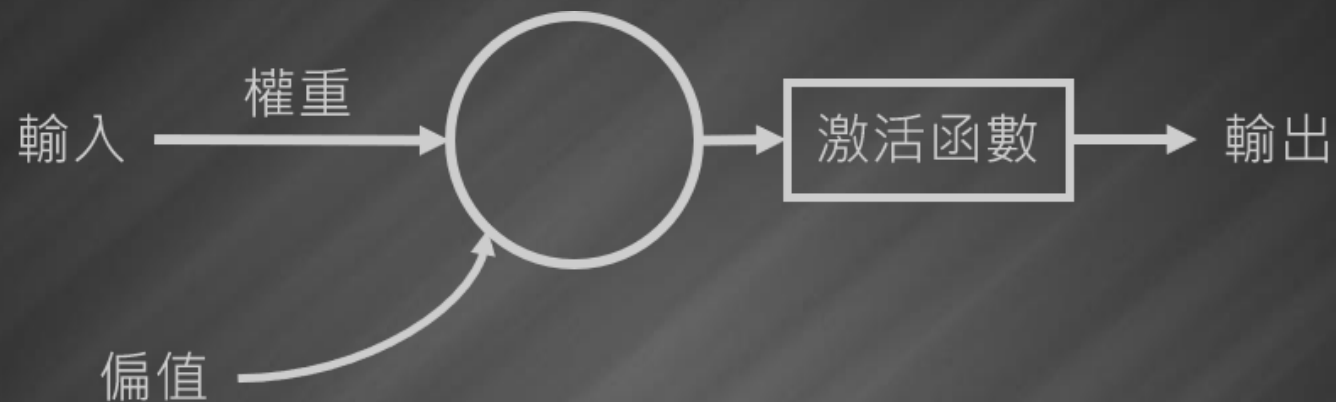
輸出 = 輸入 × 權重 + 偏值

線性函數 (又稱一次函數)

$$y = f(x) = kx + b$$

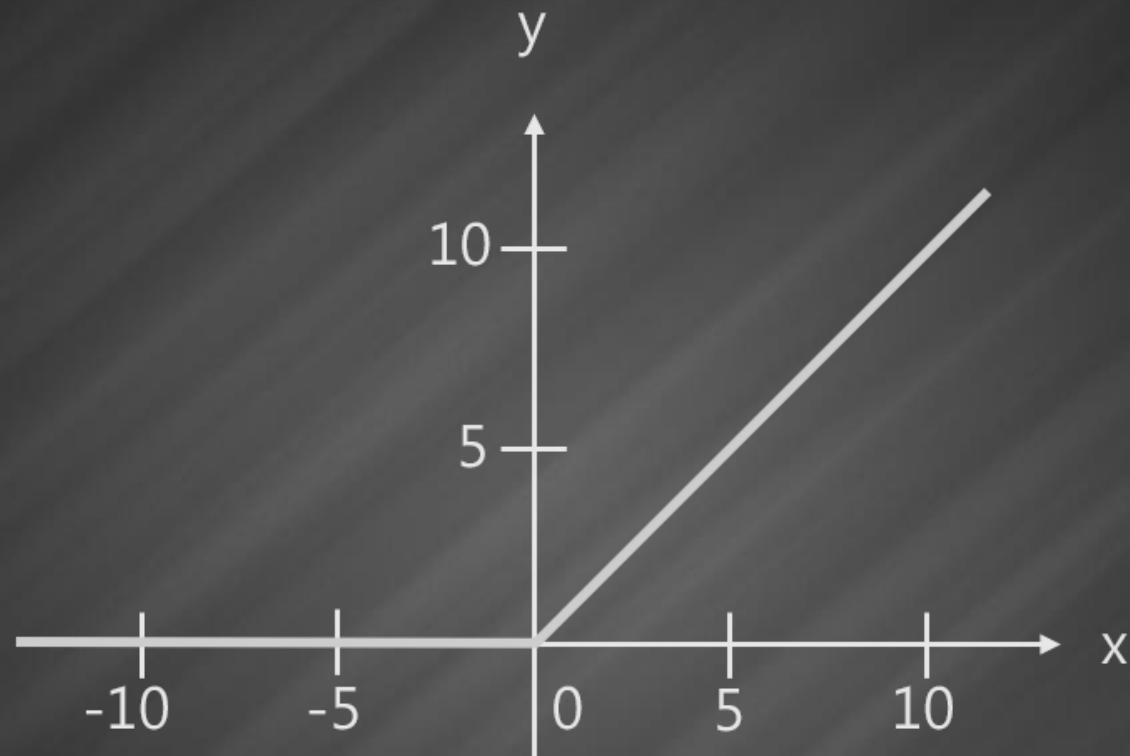


# 激活函數 (activation function)



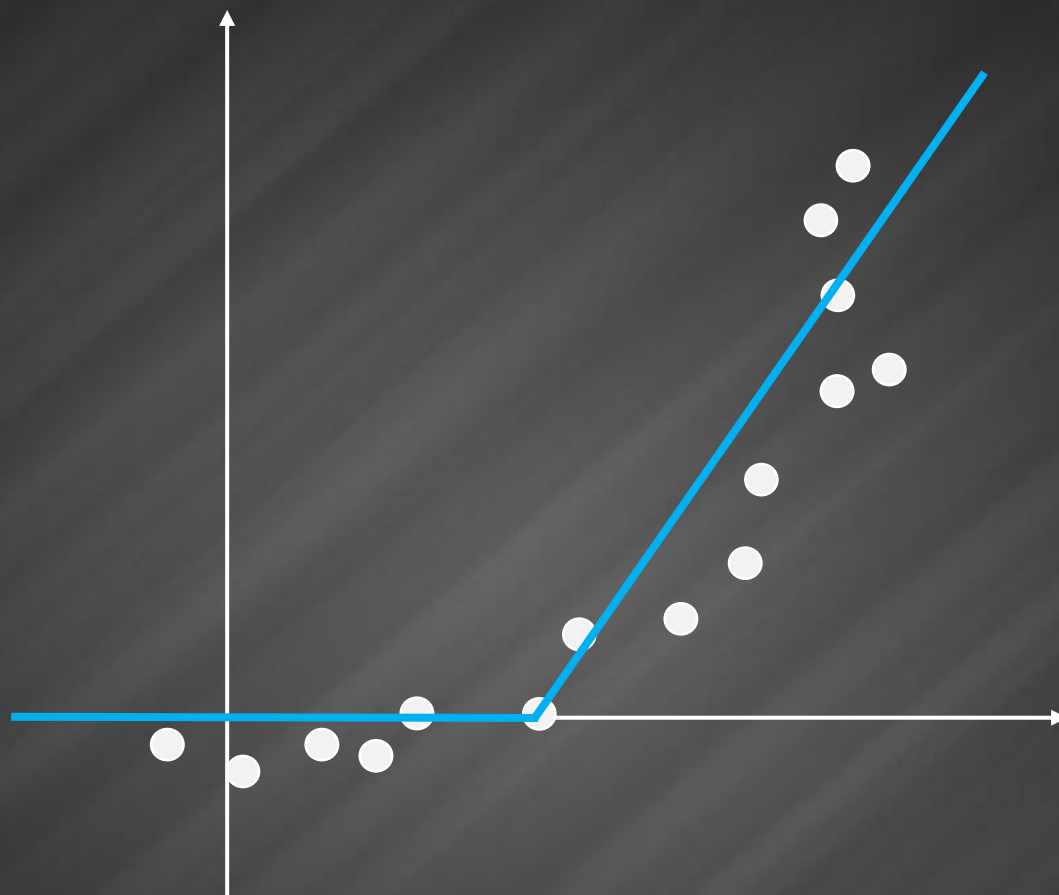
$$\text{輸出} = \text{激活函數}(\text{輸入} \times \text{權重} + \text{偏值})$$

# ReLU 函數 (線性整流函數, 增加非線性度)



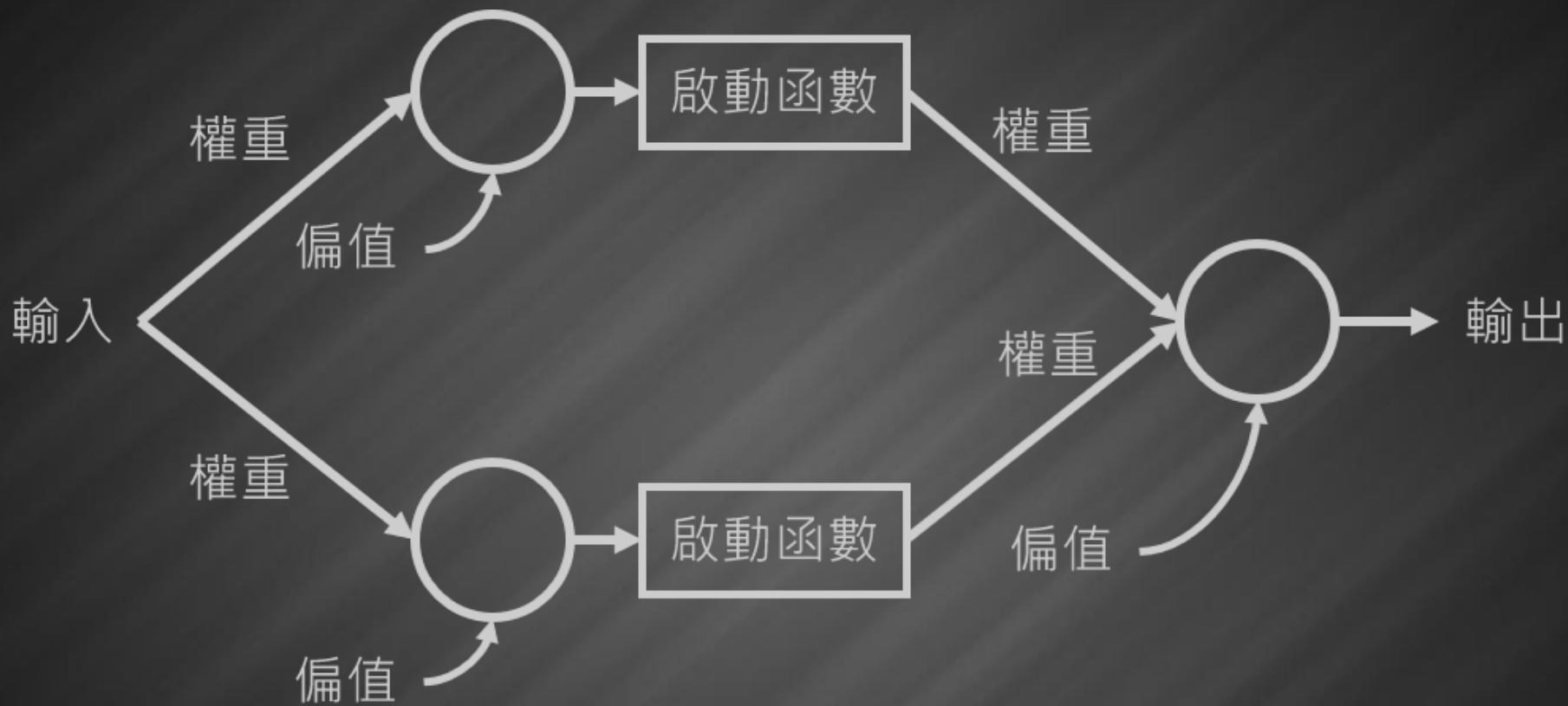
小於 0 就等於 0

# ReLU 函數 (線性整流函數, 增加非線性度)

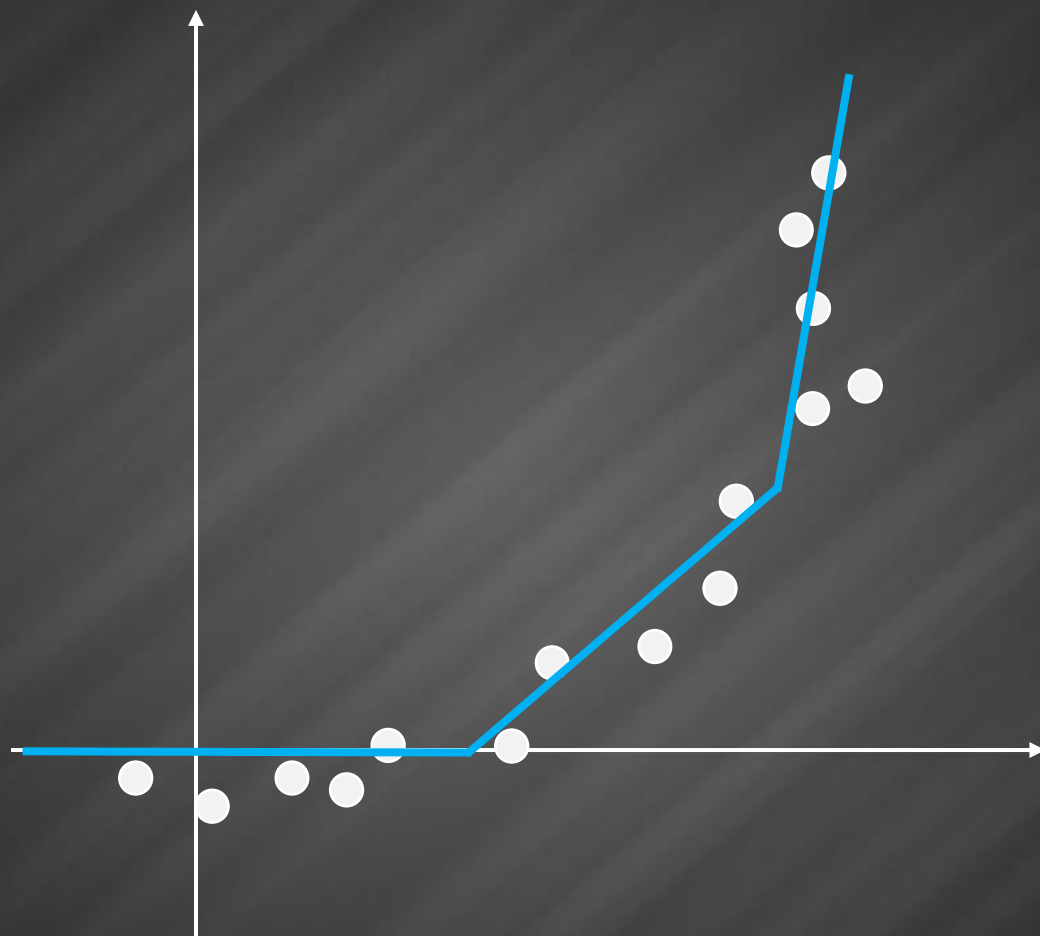


小於  $\theta$  就等於  $\theta$

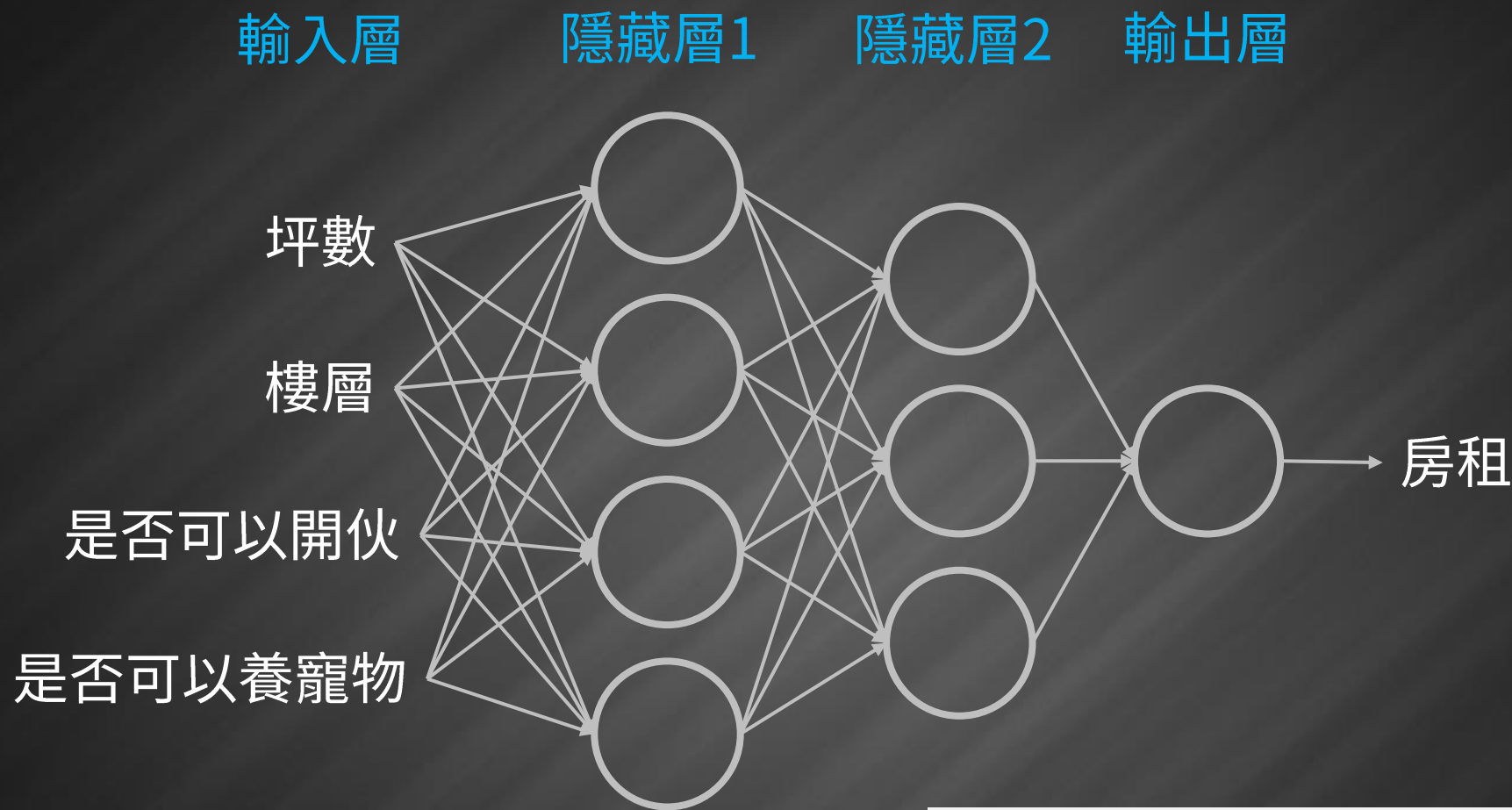
# 更貼近資料：多神經元串聯



# 更貼近資料：多神經元串聯

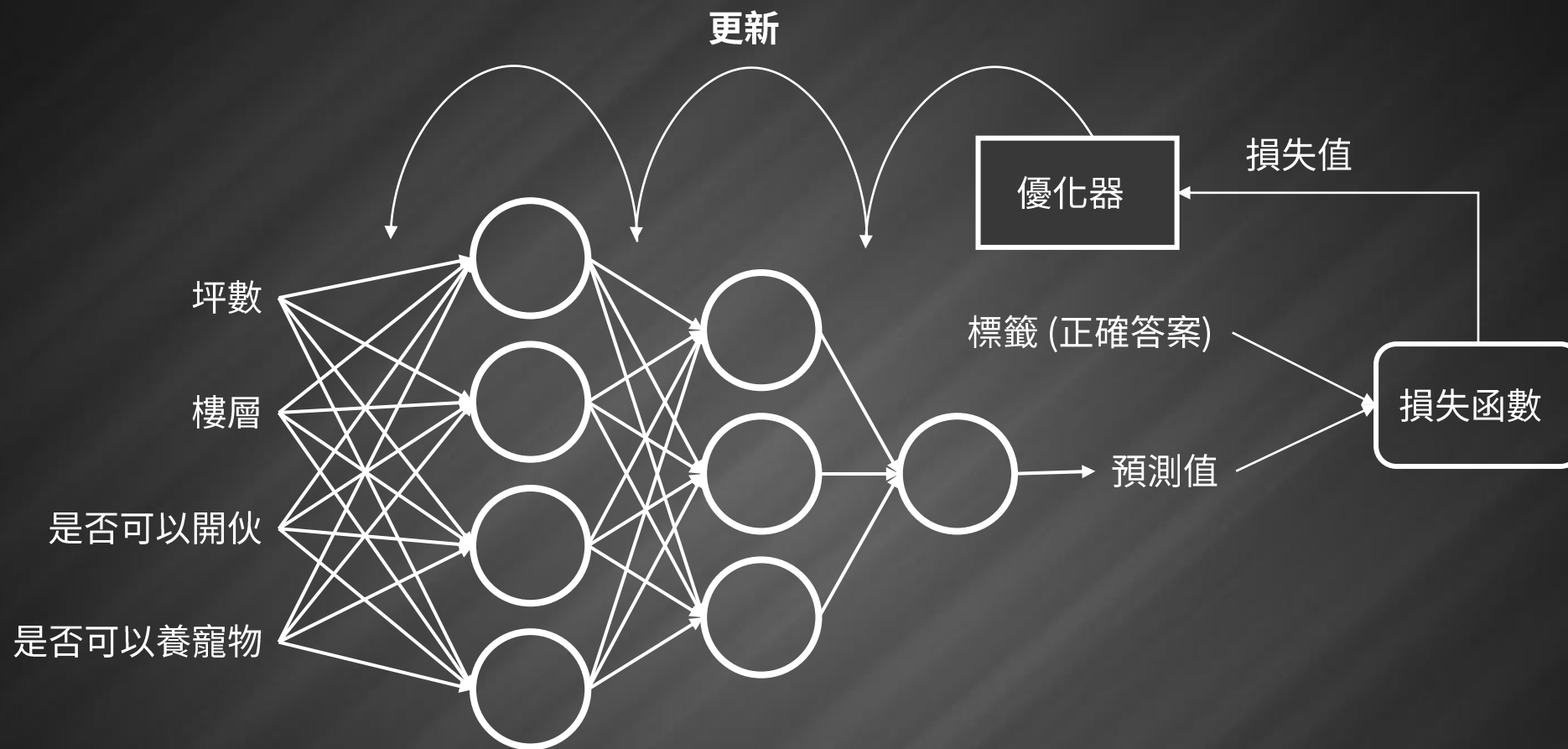


# 神經網路 (又稱為模型)



忽略偏值與激活函數，增加閱讀性

# 神經網路的學習過程



反向傳播法 (Backpropagation, BP)

# 損失函數

均方誤差 (MSE), 是將每筆標籤減掉預測值 (即誤差值) 取平方, 再取平均值。

標籤：

$$y_1、y_2、y_3、y_4、y_5 \cdots y_n$$

預測值：

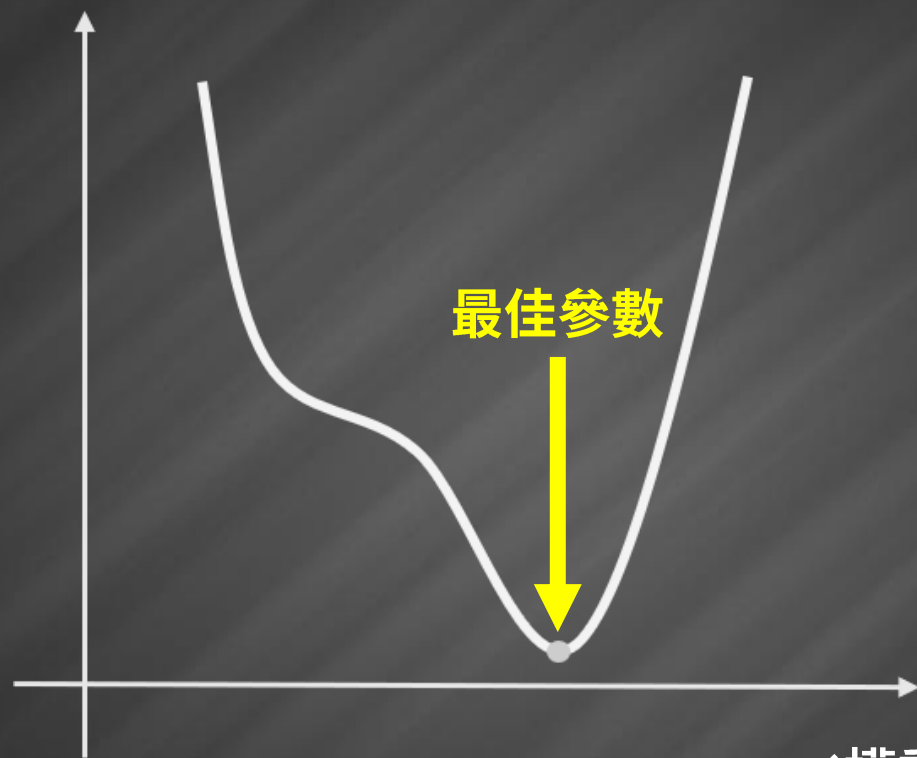
$$\hat{y}_1、\hat{y}_2、\hat{y}_3、\hat{y}_4、\hat{y}_5 \cdots \hat{y}_n$$

MSE：

$$\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

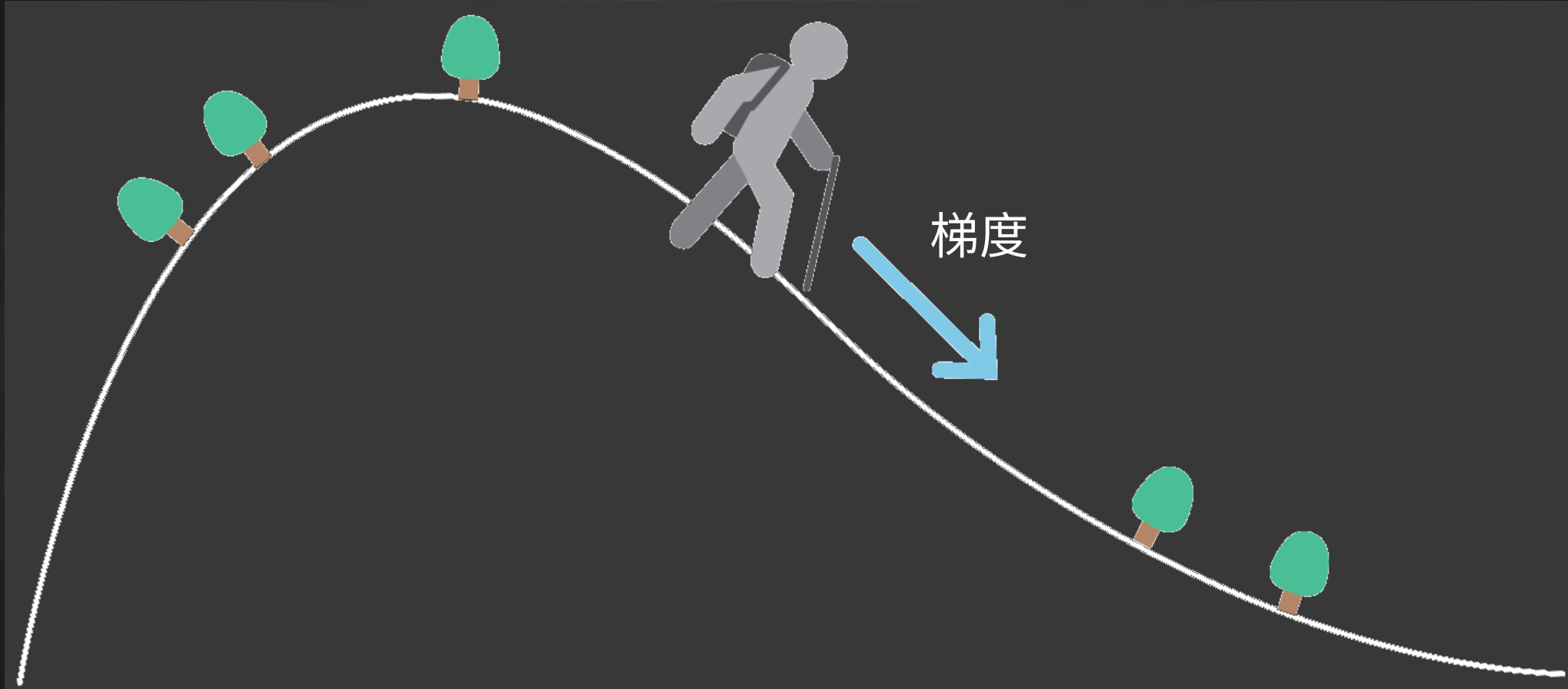
# 優化器：使用梯度下降 (Gradient descent)

loss (損失值)



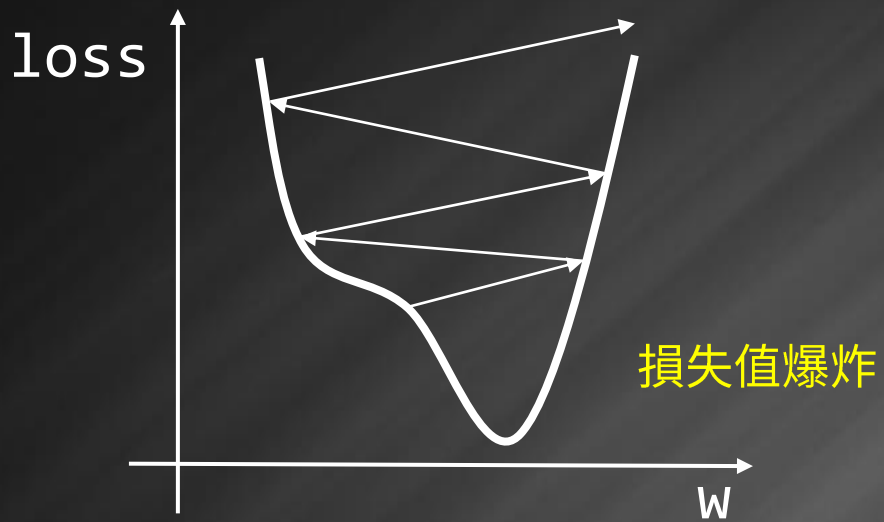
W (權重參數)

# 優化器：使用梯度下降 (Gradient descent)

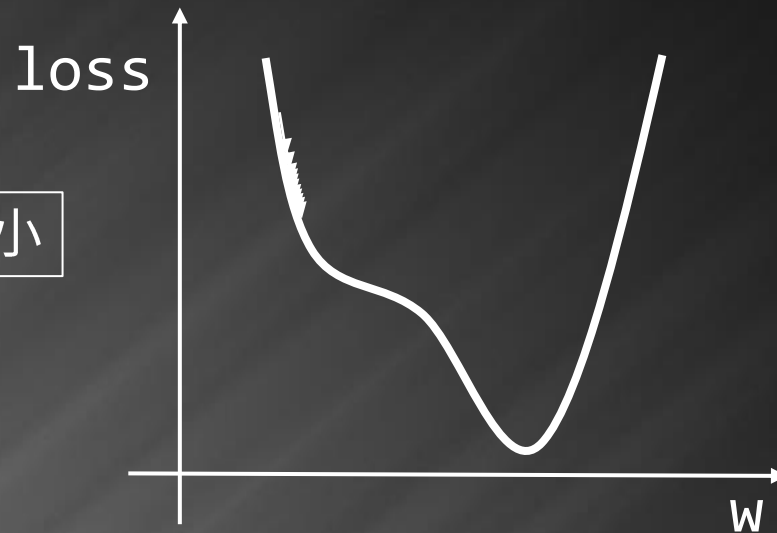


# 學習率：介於 $0 \sim 1$ (調整步伐大小)

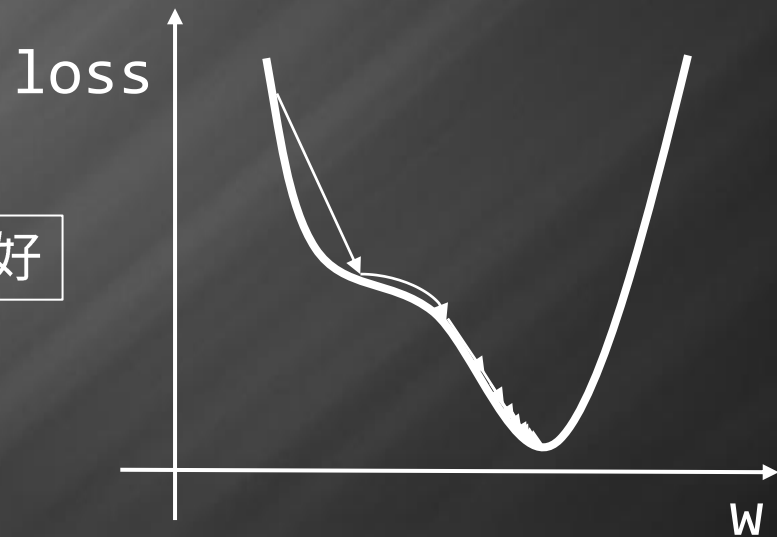
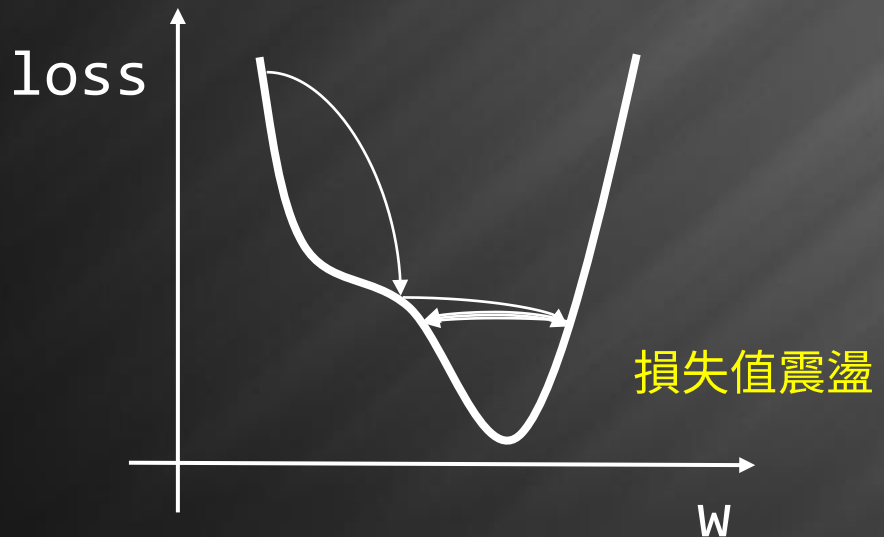
太大



太小

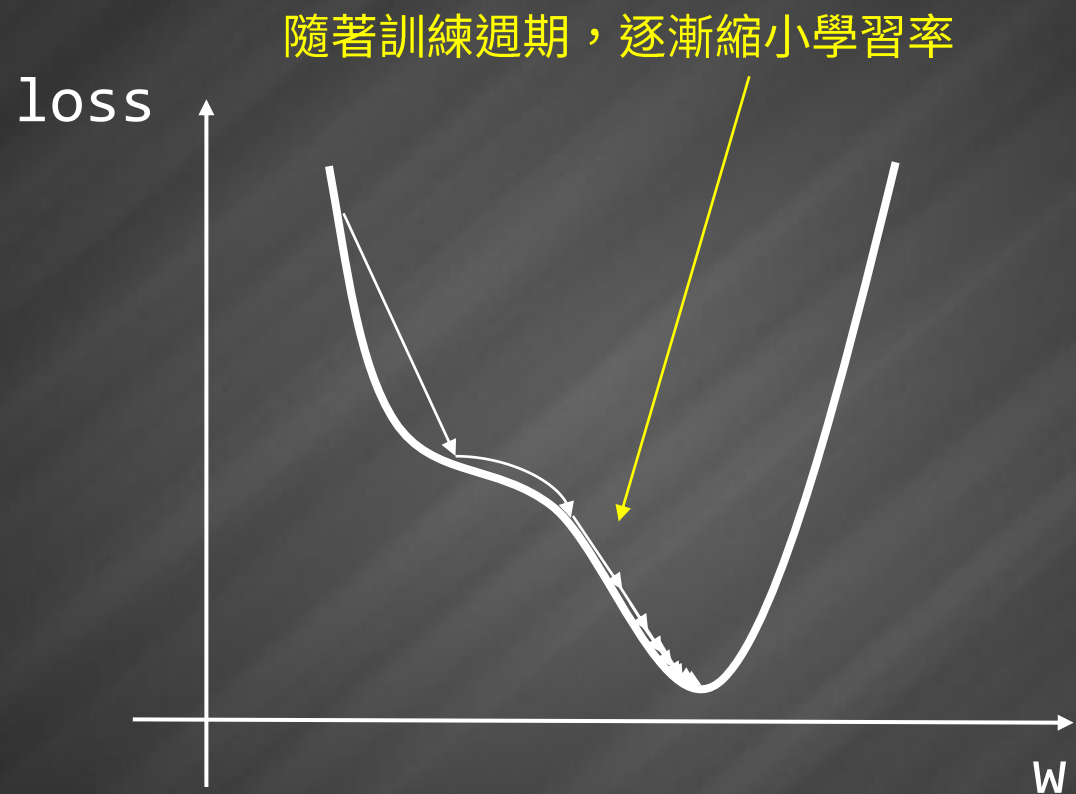


剛好



# 自適應 (Adaptive)

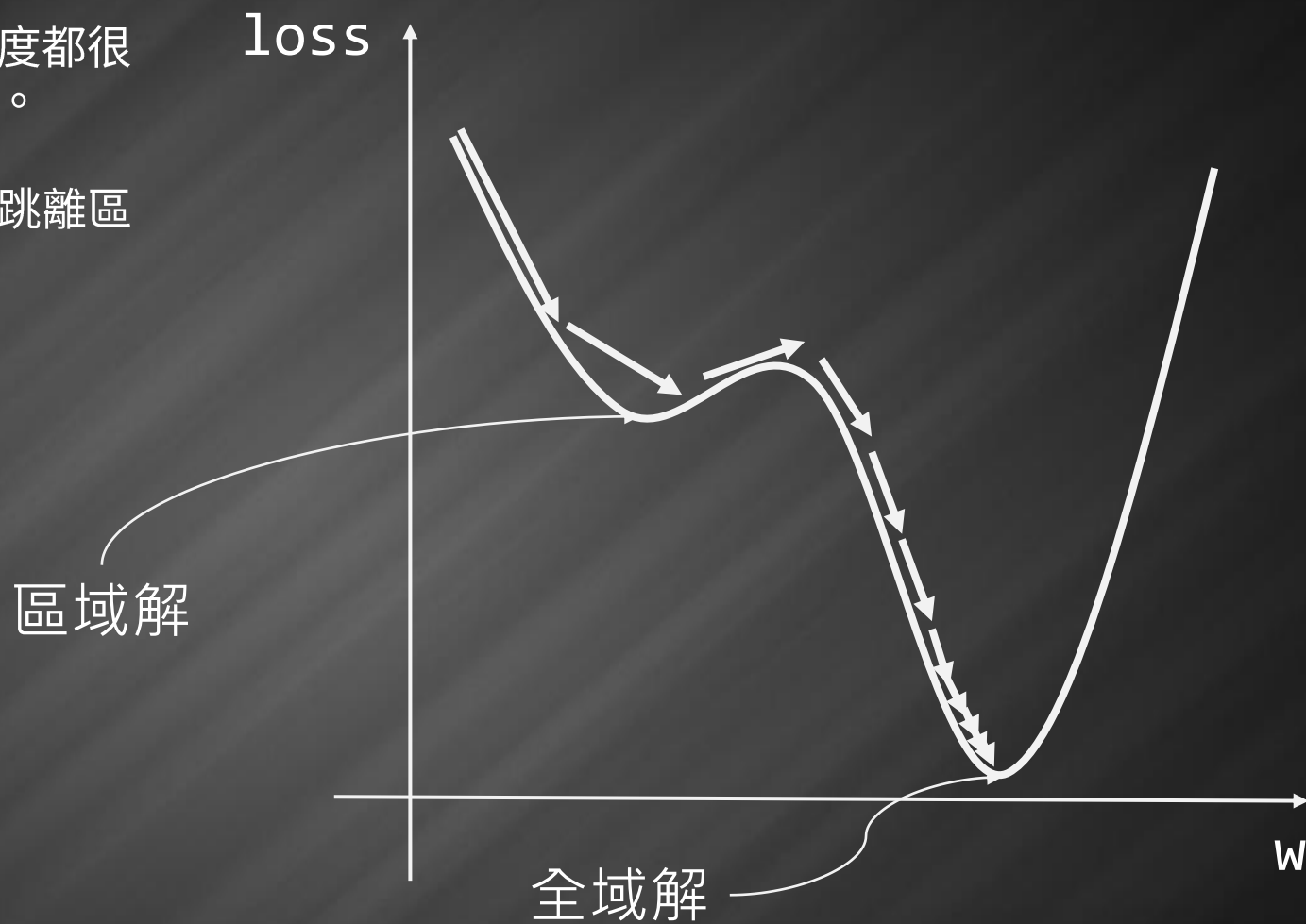
- 自適應：自動調整學習率



# 動量 (Momentum)

- 動量：解決兩個問題

1. 學習速度太慢：如果連續幾次梯度都很大，則動量可以讓移動速度加快。
2. 停留在區域最低點：加上動量，跳離區域解。



# Outline

---

- 人工智慧 (Artificial Intelligence, AI)
- 機器學習的原理
- 機器學習的 Hello World!

# 學習 AI 時的重要工具 - Python

---



# CO 免費的雲計算

---

Google  
colab

# Welcome To Colaboratory

---

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定



# 快速上手

---

 A horizontal search bar with a white background and an orange border. The text "Colab" is entered in the white field. To the right, there is an orange button containing a white magnifying glass icon.



Welcome To Colaboratory  
File Edit View Insert Runtime Tools Help

請先登入 Share Sign in

Table of contents

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

## 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
```

HTML

Markdown

Cell

# CO 快速上手 - 新建筆記本

File/New notebook

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Profile

+ Code + Text

Connect Editing

Cell : Code/Text

點此連接虛擬主機



# 快速上手 - 執程式碼

改檔名

刪除/複製/剪下Cell

執行程式碼

滑鼠移到此處可增加 Cell

CO logo

Untitled6.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

↑ ↓ ↻ 🗨 ⚙ 🗑 ⋮

▶ print('Hello Python!')

📁 Hello Python!

+ Code + Text

# Welcome To Colaboratory

---

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定



# 雲端主機的管理與設定 - GPU 加速

The screenshot shows the JupyterLab interface for a file named 'Untitled6.ipynb'. The 'Runtime' menu is open, displaying various execution options. A red box highlights the 'Runtime' menu title, and another red box highlights the 'Change runtime type' option at the bottom of the menu. A yellow box with the number '1' is placed over the 'Runtime' menu, and another yellow box with the number '2' is placed over the 'Change runtime type' option. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. The 'Runtime' menu items include: Run all (⌘/Ctrl+F9), Run before (⌘/Ctrl+F8), Run the focused cell (⌘/Ctrl+Enter), Run selection (⌘/Ctrl+Shift+Enter), Run after (⌘/Ctrl+F10), Interrupt execution (⌘/Ctrl+M I), Restart runtime... (⌘/Ctrl+M .), Restart and run all..., and Factory reset runtime. The 'Change runtime type' option is highlighted in grey. The interface also shows a code cell with the text 'print('Hello Python!')' and its output 'Hello Python!'. The top right corner features 'Comment', 'Share', and 'Settings' icons, along with a user profile picture. The bottom right corner shows RAM and Disk usage indicators and an 'Editing' mode indicator.



# 雲端主機的管理與設定 - GPU 加速

CO Untitled6.ipynb ☆ Comment Share

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

print('Hello Python')

Hello Python

### Notebook settings

Runtime type  
Python 3

Hardware accelerator  
None

Omit code cell output when saving this notebook

CANCEL SAVE

GPU

TPU

3

4

5



# 雲端主機的管理與設定 - Session 管理

The screenshot displays the JupyterLab interface for a file named 'Untitled6.ipynb'. The 'Runtime' menu is open, showing various execution options. A red box labeled '1' highlights the 'Runtime' menu header. Another red box labeled '2' highlights the 'Manage sessions' option at the bottom of the menu. In the top right corner, there is a notification box with the text '查看當前資源用量' (View current resource usage) and a profile picture. Below this, a resource usage monitor shows 'RAM' and 'Disk' usage bars, with a red box highlighting the monitor area. The main editor area contains a code cell with the text `print('Hello Python')` and its output, 'Hello Python!'.

Runtime 1

- Run all ⌘/Ctrl+F9
- Run before ⌘/Ctrl+F8
- Run the focused cell ⌘/Ctrl+Enter
- Run selection ⌘/Ctrl+Shift+Enter
- Run after ⌘/Ctrl+F10
- Interrupt execution ⌘/Ctrl+M I
- Restart runtime... ⌘/Ctrl+M .
- Restart and run all...
- Factory reset runtime
- Change runtime type
- 2 Manage sessions
- View runtime logs

查看當前資源用量

RAM  
Disk

Editing


```
print('Hello Python')
```

Hello Python!



# 雲端主機的管理與設定 - Session 管理

Active sessions

| Title  | Last execution | RAM used |                           |
|--|----------------|----------|---------------------------|
|  Untitled6.ipynb<br>Current session | 0 minutes ago  | 0.15 GB  | <a href="#">TERMINATE</a> |

點此可以關閉 Session

# Welcome To Colaboratory

---

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定



# 目錄窗格與檔案管理

開/關 目錄窗格

點此上傳本機檔案

The screenshot displays the JupyterLab interface. At the top, the title bar shows 'Untitled6.ipynb' with a star icon, and a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar, the 'Files' sidebar is visible on the left, containing an 'Upload' button (highlighted with a red box), a 'Refresh' button, and a 'Mount Drive' button (highlighted with a red box). Below these buttons, a folder named 'sample\_data' is shown. At the bottom of the sidebar, a 'Disk' progress bar indicates '79.41 GB available' (highlighted with a red box). The main code editor area shows a code cell with the text `print('Hello Python!')` and its output, 'Hello Python!'. The top right of the interface includes 'Comment', 'Share', and a user profile icon. The bottom right of the code editor shows 'RAM' and 'Disk' usage indicators, and an 'Editing' mode indicator.

虛擬主機剩餘容量

點此掛載雲端硬碟



# 目錄窗格與檔案管理 – 掛載雲端

The screenshot displays the JupyterLab interface for a notebook titled "Untitled6.ipynb". The top navigation bar includes the CO logo, the notebook title, a star icon, and buttons for "Comment", "Share", and a user profile picture. Below this is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with the status "All changes saved".

The left sidebar, titled "Files", contains a "Mount Drive" button (highlighted with a red box and a yellow "1"), "Upload", "Refresh", and a folder named "sample\_data". At the bottom of the sidebar, a "Disk" usage indicator shows "79.41 GB available".

The main workspace shows a code cell with the following content:

```
+ Code + Text  
✓ RAM [ ]  
Disk [ ]  
Editing ^  
▶ print('Hello Python!')  
↳ Hello Python!
```



# 目錄窗格與檔案管理 – 掛載雲端

The screenshot shows the top part of a Jupyter Notebook interface. The title bar includes the CO logo, the file name 'Untitled6.ipynb', and a star icon. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. To the right of the menu bar are 'Comment', 'Share', and a settings gear icon. The main interface area shows a 'Files' sidebar on the left with options for 'Upload', 'Refresh', and 'Mount Drive'. The main workspace has '+ Code' and '+ Text' buttons, a 'RAM Disk' indicator, and an 'Editing' mode button. A white dialog box is centered on the screen, asking for permission to access Google Drive files. The dialog contains the text 'Permit this notebook to access your Google Drive files?' and 'Connecting to Google Drive will permit code executed in this notebook to modify files in your Google Drive.' At the bottom of the dialog are two buttons: 'NO THANKS' and 'CONNECT TO GOOGLE DRIVE'. The 'CONNECT TO GOOGLE DRIVE' button is highlighted with a red border. A yellow box with the number '2' is placed above the 'CONNECT TO GOOGLE DRIVE' button.

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved Comment Share ⚙️

Files × + Code + Text ✓ RAM Disk Editing ^

⬆️ Upload ↻ Refresh 🗄️ Mount Drive

<>

📁

**Permit this notebook to access your Google Drive files?**

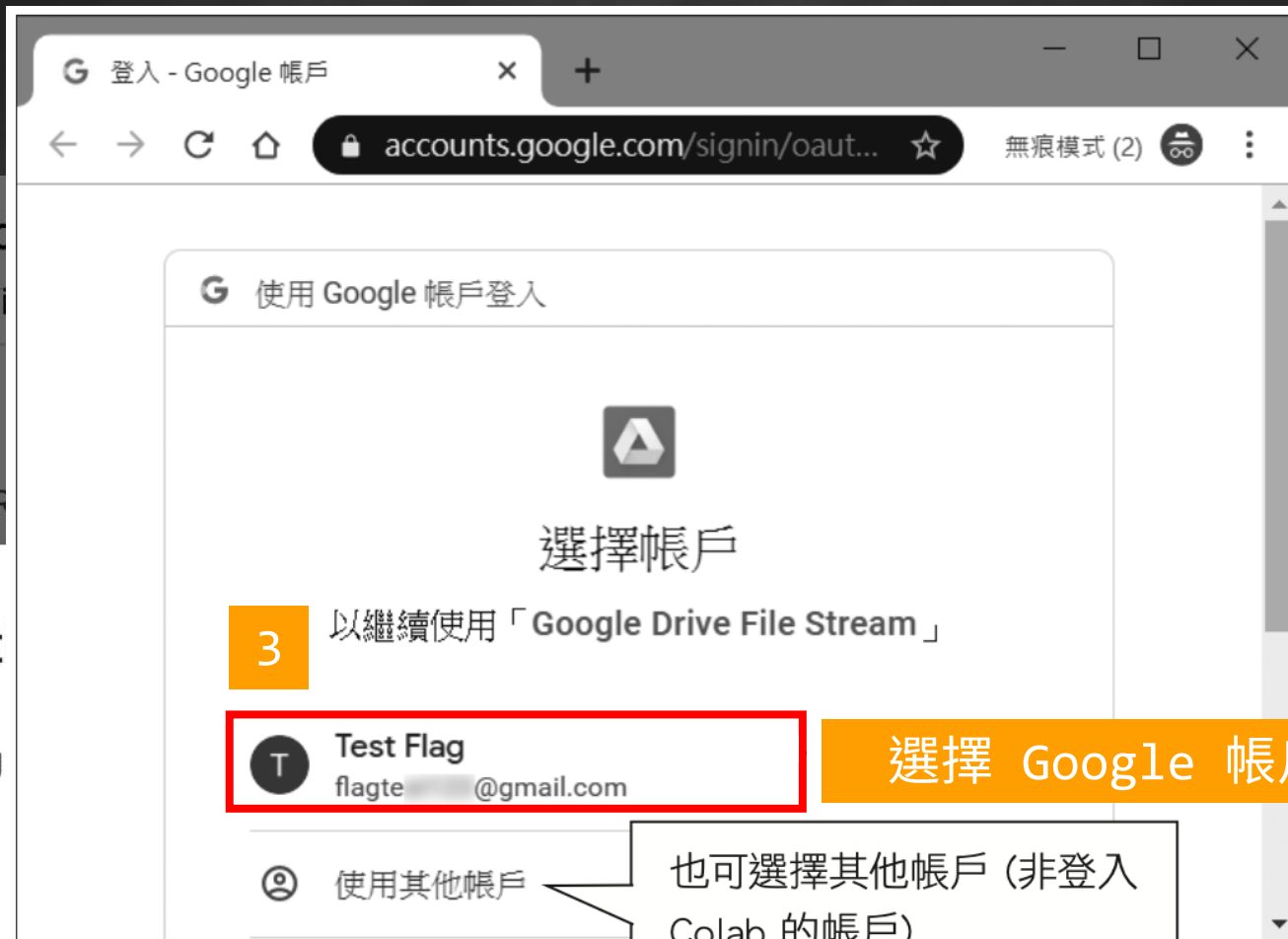
Connecting to Google Drive will permit code executed in this notebook to modify files in your Google Drive.

2

NO THANKS **CONNECT TO GOOGLE DRIVE**



# 目錄窗格與檔案管理 - 掛載雲端



Permit t  
Connecting

選擇 Google 帳戶  
Google Drive.  
GOOGLE DRIVE

也可選擇其他帳戶 (非登入 Colab 的帳戶)



# 目錄窗格與檔案管理 – 掛載雲端

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved Comment Share Settings Profile

RAM Disk

+ Code + Text

Editing

Files

Upload Refresh Unmount Drive

drive

sample\_data

20...

print('Hello Python!')

Python!

Download

Delete file

Rename file

Copy path

Refresh

掛載成功後會看到此資料夾

對檔案按右鍵可取得路徑



# 目錄窗格與檔案管理 – 線上解壓縮

The screenshot shows a Jupyter Notebook interface with the following elements:

- Header:** Untitled6.ipynb, File Edit View Insert Runtime Tools Help, All changes saved, Comment, Share, and a user profile icon.
- Files Panel (Left):** Shows a file tree with folders 'drive' and 'sample\_data', and files '20191104\_labelme' and '20191104\_labelme.zip'. The file '20191104\_labelme' is highlighted with a red box, and '20191104\_labelme.zip' is also highlighted with a red box.
- Code Cell (Right):** Contains the following code:

```
[1] print('Hello Python!')
```

```
!unzip /content/20191104_labelme.zip
```

```
ve: /content/20191104_labelme.zip
```

```
eating: 20191104_labelme/
```

```
104_labelme/2017_PAS_400_json/
```

```
acing: 20191104_labelme/2017_PAS_400_json/101.01
```

```
lating: 20191104_labelme/2017_PAS_400_json/101.01
```

```
lating: 20191104_labelme/2017_PAS_400_json/201701
```

```
lating: 20191104_labelme/2017_PAS_400_json/201701
```

```
eating: 20191104_labelme/2018_PAS_400_json/
```

Four orange callout boxes provide instructions:

1. 按右鍵取得路徑 (Right-click to get the path)
2. 利用 linux 指令和剛剛複製的路徑 (Use linux command and the path just copied)
3. 執行此 Cell (Execute this Cell)
4. 解壓縮成功 (Unzip successful)

# Welcome To Colaboratory



---

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定



# 偏好設定

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved Comment Share  

Files

Upload Refresh Unmount

- ..
- 20191104\_labelme
- drive
- sample\_data
- 20191104\_labelme.zip

Command palette

Settings...

Keyboard shortcuts... ⌘/Ctrl+M H !')

```
!unzip /content/20191104_labelme.zip
```

這兩個地方皆可進入設定頁面



# 偏好設定 - 主題設定 (暗黑)

**Settings**

**Site**

Editor

Colab Pro

Miscellaneous

Theme  
light

New notebooks use private outputs (omit outputs when saving)

Request GitHub access to view and edit private repositories and organizations

[More info](#)

Custom snippet notebook URL

CANCEL SAVE

Disk 79.38 GB available



# 偏好設定 - 主題設定 (暗黑)

The screenshot shows the 'Settings' dialog box in the CO application. The 'Site' category is selected in the left sidebar. The 'Theme' section is active, showing a dropdown menu with three options: 'dark', 'light', and 'adaptive'. The 'dark' option is highlighted with a red border and a yellow '2' in a box, indicating it is the selected theme. Below the theme selection, there are two unchecked checkboxes: 'New notebooks use...' and 'Request GitHub authentication for private repositories and organizations'. A 'More info' link is also present. At the bottom of the dialog, there are 'CANCEL' and 'SAVE' buttons. The background shows a file explorer sidebar and a status bar at the bottom indicating 'Disk 79.38 GB available'.

Settings

Site

Editor

Colab Pro

Miscellaneous

Theme

dark light

New notebooks use... (commit outputs when saving)

Request GitHub authentication for private repositories and organizations

[More info](#)

adaptive

Custom snippet notebook URL

CANCEL SAVE

Disk 79.38 GB available



# 偏好設定 - 柯基犬與小貓模式

Settings

Site

Editor

Colab Pro

**Miscellaneous** 3

Power level  
Some power

Corgi mode 4

Kitty mode

CANCEL SAVE 5



# 偏好設定 - 柯基犬與小貓模式

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comments Share Settings Profile

Files

Upload Refresh Unmount Drive

- ..
- 20191104\_labelme
- drive
- sample\_data
- 20191104\_labelme.zip

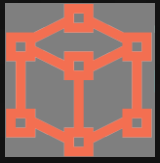
+ Code + Text RAM Disk Editing

```
[1] print('Hello Python!')
```

```
↳ Hello Python!
```

```
!unzip /content/20191104_labelme.zip
```

Disk 79.38 GB available



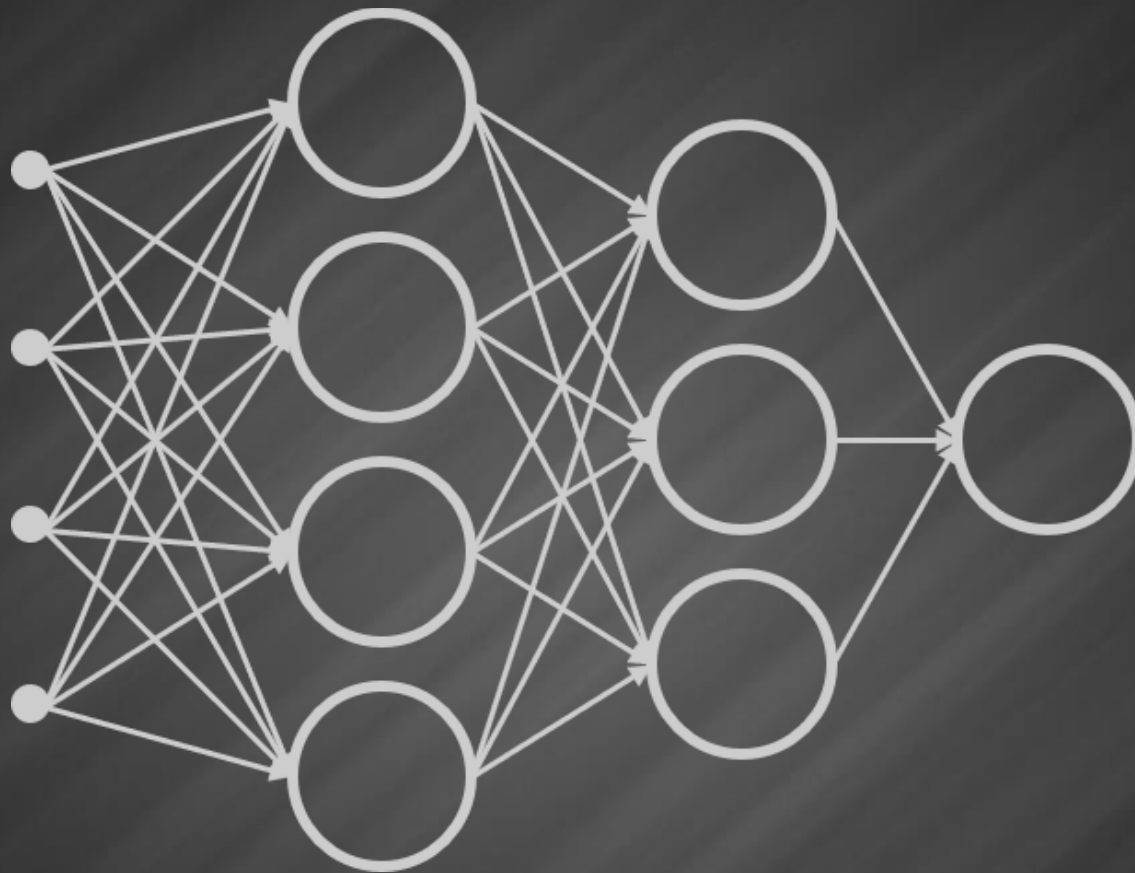
# Keras 基本介紹

---

- 建立神經網路像是堆積木一樣，簡單明瞭。
- 支援處理影像辨識、文字..，等內容的神經網路（ex：CNN、RNN）。
- 程式碼在更換硬體環境時（CPU、GPU），無須做任何更改。

# 用 Keras 建構神經網路

---



# 建立空的神經網路模型

# 匯入 Keras 的序列式模型類別

```
from tensorflow.keras.models import Sequential
```

# 匯入 Keras 的密集層類別

```
from tensorflow.keras.layers import Dense
```

# 建立神經網路

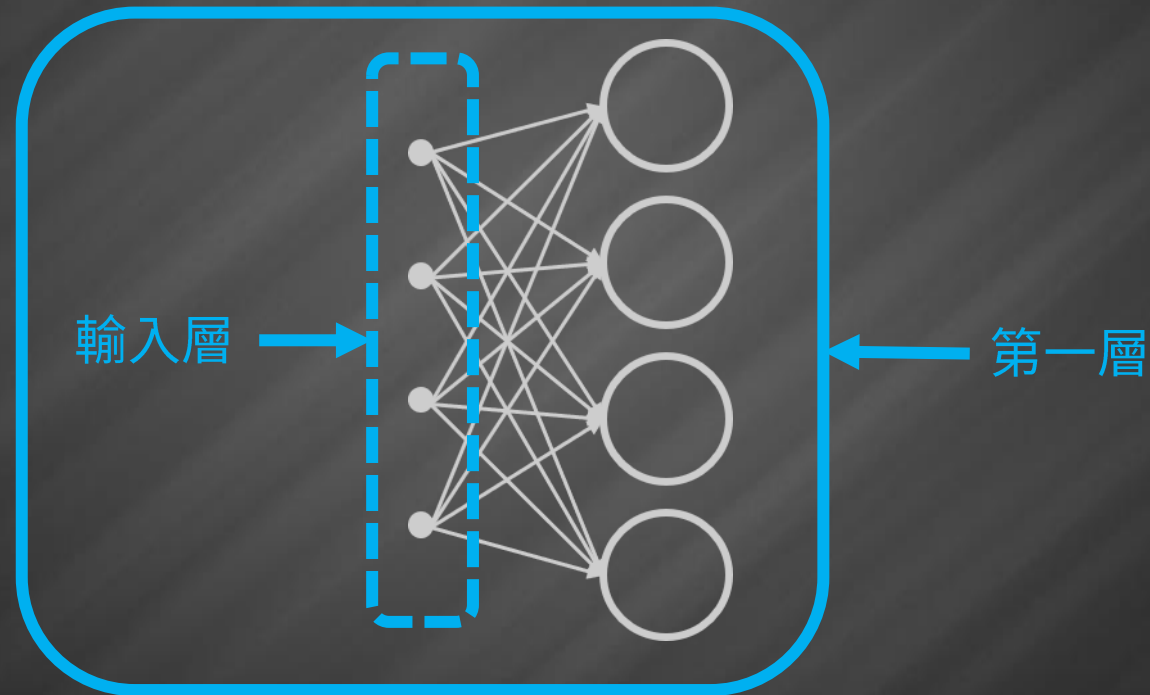
```
model = Sequential()
```

← 建立序列模型物件，並指定給 `model` 變數，這時的 `model` 就是一個神經網路了，但內容是空的

# 加入第一層的神經層（包含輸入層功能）

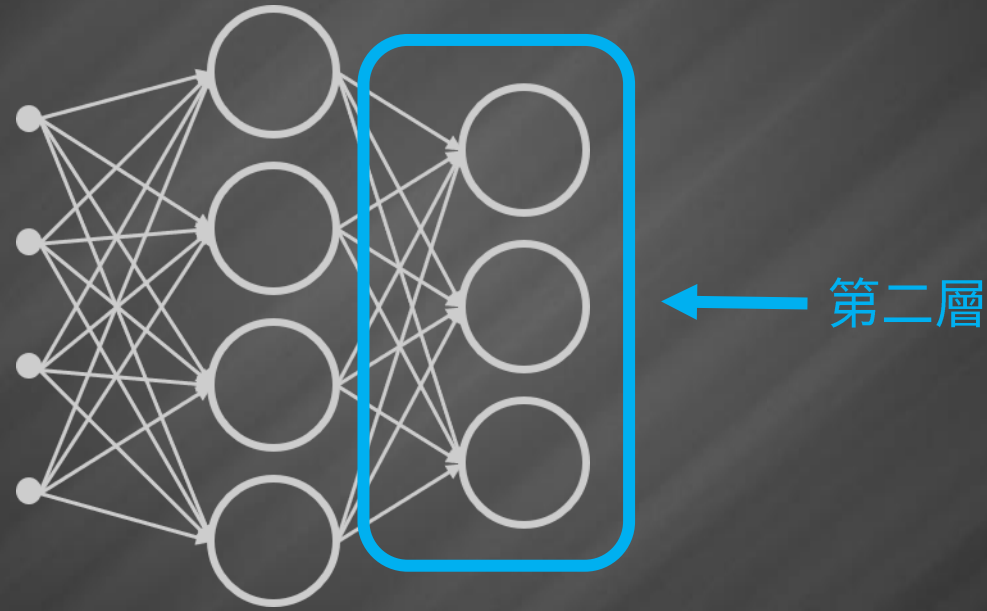
```
model.add(Dense(4, activation='relu', input_shape= (4, ))) ← 輸入層形狀
```

密集層 (Dense layer) 是最普通的神經層，它的每一個神經元都會與上一層的每個神經元連接，又稱為全連接層



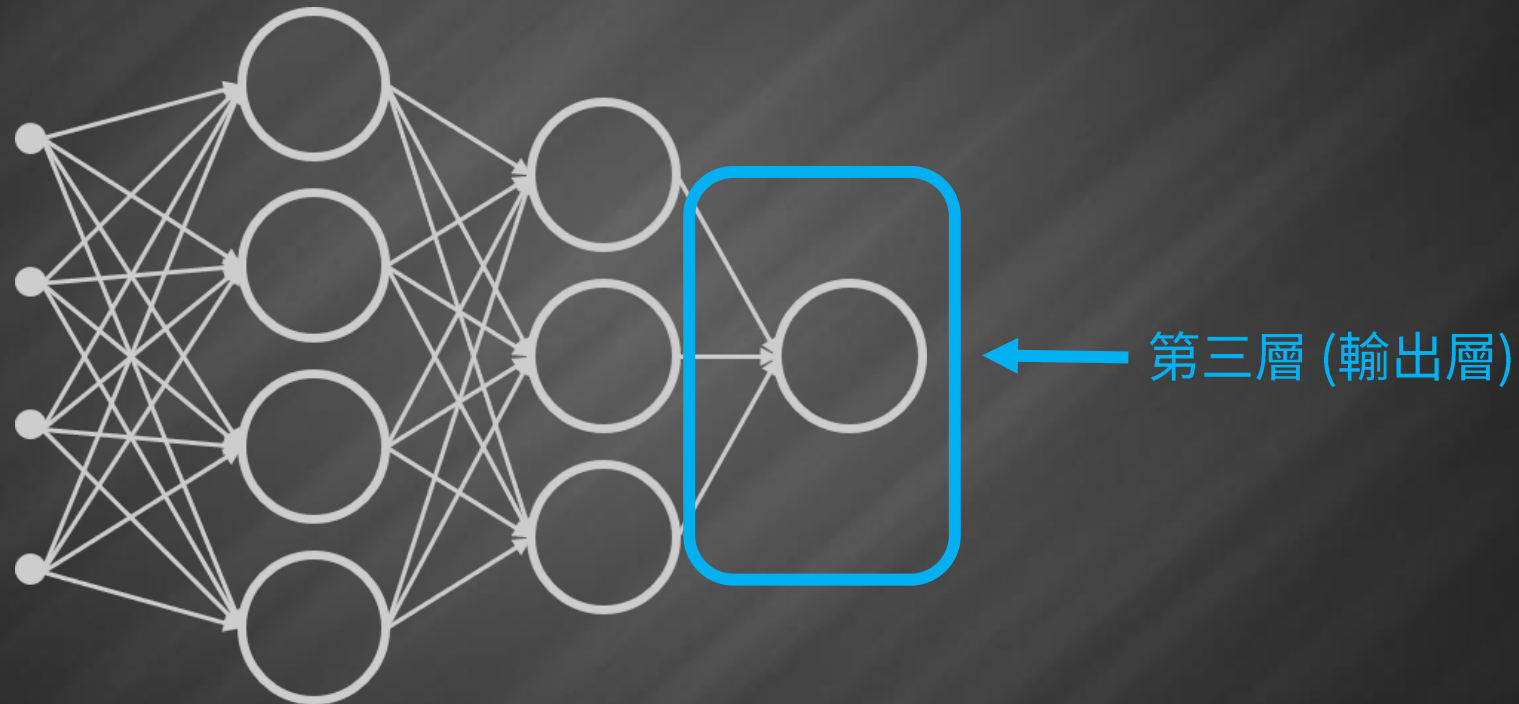
# 加入第二層的神經層

```
model.add(Dense(3, activation='relu')) ← 除第一層之外都不用指定  
input_shape 參數
```



# 加入輸出層

`model.add(Dense(1))` ← 再加入一個密集層，只有 1 個神經元，並且不使用激活函數



# 顯示當前模型架構

exA-3

```
model.summary() ← 顯示當前模型架構及參數
```

# LAB01 預測台灣房租

實驗目的

利用神經網路的迴歸模型來預測『台北市中山區』房租。

材料

無

開發環境

Colab

# 資料介紹

---

- 此資料集蒐集自『591 房屋交易網』，地點為『台北市中山區』，每筆資料蒐集了『坪數』、『樓層』、『是否可以開伙』和『是否可以養寵物』共 4 種特徵，並有對應的『房租價格』。
  - ✓ 總資料數：689
  - ✓ 特徵資料數量：4
  - ✓ 標籤資料數量：1（房租價格）

# 上傳房屋資料、第三方模組

```
# 匯入「房屋txt檔」和『第三方函式庫』到 Colab
from google.colab import files

uploaded = files.upload() # 匯入房屋 .txt 檔
uploaded = files.upload() # 匯入第三方函式庫 keras_lite_convertor
```

# 讀取檔案

```
# 讀取 house.txt 檔案，並得出特徵和標籤
import keras_lite_convertor as kc

path_name = 'house.txt' # 檔案路徑
Data_reader = kc.Data_reader(path_name, mode = 'regression') # 指
定讀檔模式 (regression 適用於迴歸預測)
data, label = Data_reader.read(random_seed = 12) # 將檔案讀到
的 5 種資料分為『特徵』和『標籤』，並設定亂數種子為 12 (data, label 為
numpy.ndarray 格式)
```

物件一定有資料型態，可用 `type(物件或變數名稱)` 查詢。例如：

- `type(data)`：會回傳 `numpy.ndarray`。

等號左邊是指定變數，不會顯示物件內容。可用變數名稱或 `print(變數名稱)`，來檢查物件的內容。

- `print(data)`：會回傳 `numpy` 物件的內容。

# Python 的資料結構 (容器)

# Python 的基本資料結構

- 字串容器：由字元組成。例如：`string = "52python"`。
- tuple 容器：由資料物件組成。例如：`tuple = (1, (2, ), 3)`。
- 串列容器：由資料物件組成。例如：`list = [1, [2], 3]`。
- 集合容器：由資料物件組成。例如：`set = {1, '2', 3}`。
- 字典容器：由資料物件組成，以鍵：值表示。例如：`dick = {'A':1, 'B':'2', 'C':3}`。

# Python 的第三方資料結構：numpy.array

- Numpy：Python 的擴充模組，常用於機器學習的資料處理。

```
import numpy as np                    # 匯入 numpy
a = np.array([10, 2, 45, 32, 24])     # 建立五個元素

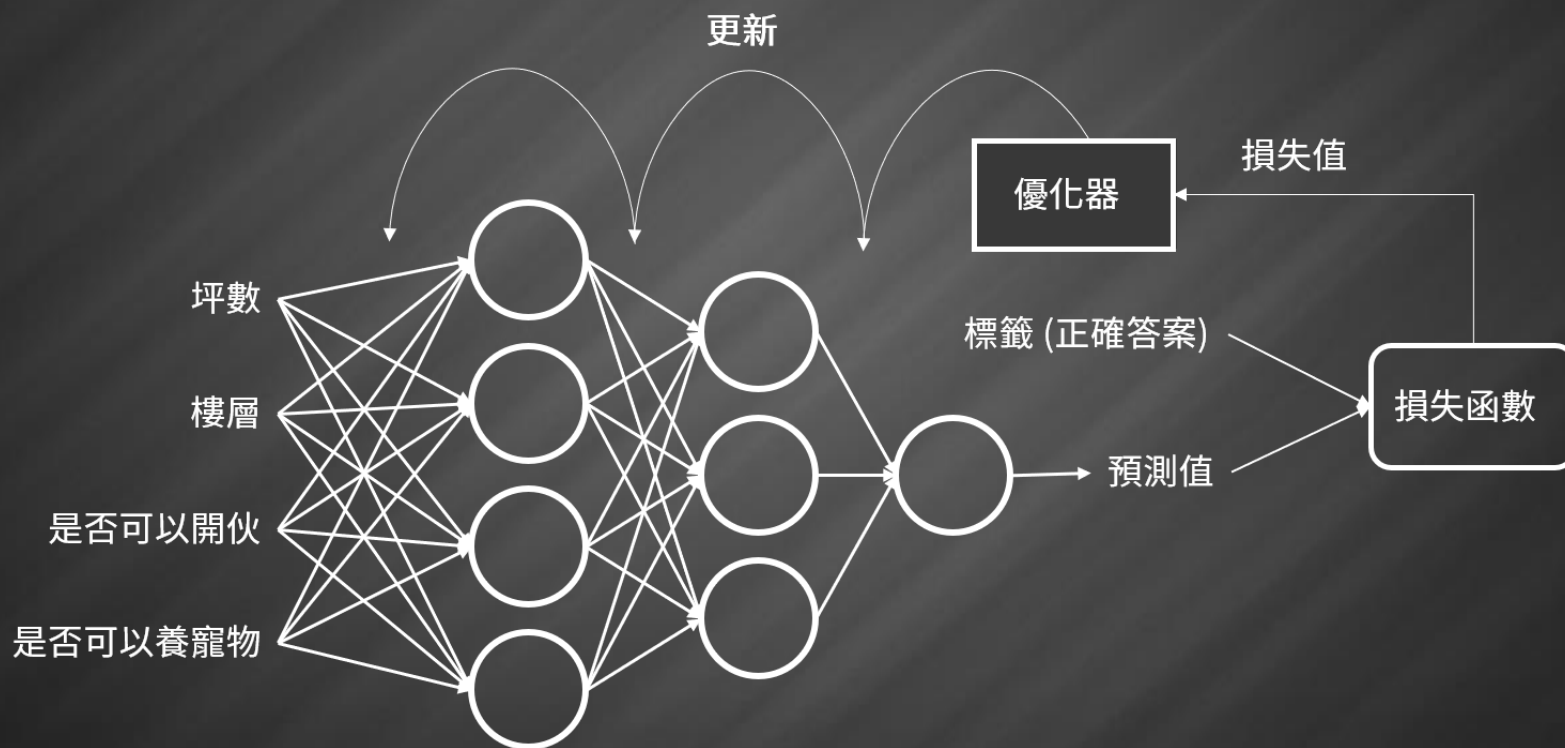
>>> len(a)                            # len() 會回傳容器內元素的數量

>>> a[2:4]                             # 索引取值 (位置 2 ~ 3)

>>> a[:4]                              # 索引取值 (位置 0 ~ 3)
```

# 訓練集、驗證集、測試集

- 訓練集：神經網路訓練時只會看到訓練集。(就像學生在學習時，寫的例題)
- 驗證集：訓練過程使用驗證集來模擬測試。(就像學生的習題)
- 測試集：訓練完畢，用測試集來考他。(就像學生的期末考試)



```
# 資料預處理  
# 取資料中的 90% 當作訓練集  
split_num = int(len(data) * 0.9)  
train_data = data[:split_num]  
train_label = label[:split_num]
```

容器的元素個素，可用 `len(容器的變數名稱)` 查詢。例如：

- `len(train_data)`：會回傳 620。
- `len(train_label)`：會回傳 620。

# 資料正規化

- 每種特徵值的屬性和範圍都不太一樣，例如：
  - ✓ 坪數：範圍介於 4 ~ 26 坪。
  - ✓ 是否可以養寵物：只有 0 與 1。
- 這會導致數值越大，對權重的影響也越大，解決方式：
  - ✓ 讓每種特徵使用相同的計量標準。
- 訓練集的特徵：先將資料減掉平均，再將其除以標準差。  
(以 0 作為基準，標準差作為單位)
- 訓練集的標籤：除以標籤的最大值。(落在 0 ~ 1 之間)

# 標準差的計算

- 例如：一群孩童年齡的數值為  $\{5, 6, 8, 9\}$ 
  - ✓ 第一步：計算平均值

$\bar{x}$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

$n = 4$  (因為集合里有 4 個數)，分別設為： $x_1 = 5, x_2 = 6, x_3 = 8, x_4 = 9$

$$\bar{x} = \frac{1}{4} \sum_{i=1}^4 x_i \text{ 用 } 4 \text{ 取代 } N$$

$$\bar{x} = \frac{1}{4} (x_1 + x_2 + x_3 + x_4)$$

$$\bar{x} = \frac{1}{4} (5 + 6 + 8 + 9)$$

$\bar{x} = 7$  此為平均值。

# 標準差的計算

## ✓ 第二步：計算標準差

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$$

$$\sigma = \sqrt{\frac{1}{4} \sum_{i=1}^4 (x_i - \bar{x})^2} \text{ 用 } 4 \text{ 取代 } N$$

$$\sigma = \sqrt{\frac{1}{4} \sum_{i=1}^4 (x_i - 7)^2} \text{ 用 } 7 \text{ 取代 } \bar{x}$$

$$\sigma = \sqrt{\frac{1}{4} [(x_1 - 7)^2 + (x_2 - 7)^2 + (x_3 - 7)^2 + (x_4 - 7)^2]}$$

$$\sigma = \sqrt{\frac{1}{4} [(5 - 7)^2 + (6 - 7)^2 + (8 - 7)^2 + (9 - 7)^2]}$$

$$\sigma = \sqrt{\frac{1}{4} [(-2)^2 + (-1)^2 + 1^2 + 2^2]}$$

$$\sigma = \sqrt{\frac{1}{4} (4 + 1 + 1 + 4)}$$

$$\sigma = \sqrt{\frac{10}{4}}$$

$$\sigma = 1.5811$$

# 標準差的計算

- {5, 6, 8, 9}

- ✓ 平均值：7

- ✓ 標準差：1.5811

- ✓ 先將資料減掉平均，再將其除以標準差

- $5 - 7 = -2, -2/1.5811 = -1.2649$

- $6 - 7 = -1, -1/1.5811 = -0.6324$

- $8 - 7 = 1, 1/1.5811 = 0.6324$

- $9 - 7 = 2, 2/1.5811 = 1.264$

- {15, 16, 18, 19}

- ✓ 平均值：17

- ✓ 標準差：1.5811

- ✓ 先將資料減掉平均，再將其除以標準差

- $15 - 17 = -2, -2/1.5811 = -1.2649$

- $16 - 17 = -1, -1/1.5811 = -0.6324$

- $18 - 17 = 1, 1/1.5811 = 0.6324$

- $19 - 17 = 2, 2/1.5811 = 1.264$

```
# 正規化
mean = train_data.mean()    # 平均數
data -= mean
std = train_data.std()      # 標準差
data /= std
```

```
# 將 label 範圍落在 0 ~ 1 (label 正規化)  
New_label = label / max(label)
```

# 訓練集、驗證集、測試集的資料形狀

---

- house.txt 有 689 筆資料
  - ✓ 訓練集：佔 90% (620 筆)。
  - ✓ 驗證集：10% 減掉測試集，剩下的當驗證集 (39 筆)。
  - ✓ 測試集：10% 中的最後 30 筆。

# 訓練集、驗證集、測試集的資料形狀

```
# 訓練集、驗證集、測試集的資料形狀
# 訓練集
train_data = data[:split_num] # 訓練用資料 (620 筆)
print(train_data.shape)
train_label = new_label[:split_num] # 訓練用標籤 (620 筆)
# 驗證集
validation_data = data[split_num:-30] # 驗證用資料 (39 筆)
print(validation_data.shape)
validation_label = new_label[split_num:-30] # 驗證用標籤 (39 筆)
# 測試集
test_data = data[-30:] # 測試用資料 (30 筆)
print(test_data.shape)
test_label = new_label[-30:] # 測試用標籤 (30 筆)
```

# 建立神經網路架構

- 建立幾層，以及每層多少神經元，只能透過經驗或不段測試。
  - ✓ 建立一個含輸入層共 3 層的神經網路，其中兩個隱藏層皆設定為 20 個神經元。

```
# 建立神經網路架構
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential() # 建構網路模型
# 增加一層神經層，使用 ReLU 激活函數，輸入層有 4 個輸入特徵
model.add(Dense(20, activation = 'relu', input_shape = (4,)))

# 增加一層神經層，使用 ReLU 激活函數
model.add(Dense(20, activation = 'relu'))
model.add(Dense(1)) # 增加輸出為 1 的輸出層
```

# 編譯及訓練模型

- 回歸問題，使用均方誤差，且設定優化器為 "adam"。
  - ✓ Adam 具備了不錯的自適應與動量，來尋找最佳權重。

```
# 編譯及訓練模型
```

```
# 編譯模型
```

```
model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])  
history = model.fit(train_data, train_label, validation_data = (validation_data,  
validation_label), epochs = 200)
```

- `optimizer = 'adam'`：設定最優化方法為 adam。
- `loss = 'mse'`：設定損失函式為均方誤差 mse。
- `metrics = ['mae']`：設定評估模型方式 mae 準確率。
- `epochs = 200`：訓練次數為 200 次。

# 查看損失值

- 將損失值的曲線顯示出來，確認神經網路有往目標前進。

```
# 查看損失值
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], "r", label = 'loss')
plt.plot(history.history['val_loss'], "b", label = 'val loss')
plt.legend()          # 顯示標籤
plt.show()           # 顯示圖片
```

- `loss`：使用訓練資料，得到的損失函式誤差值（值越小準確率越高）。
- `val_loss`：使用驗證資料，得到的損失函式誤差值（值越小準確率越高）。

```
# 資料比較圖
import numpy as np

plt.figure(figsize = (10, 8))          # 定義一個視窗(10,8 為視窗大小)
plt.subplots_adjust(hspace = 0.3)     # 調整兩張圖的間距
# 實際值-預測值(* max(label) 表示恢復原始值)
error = test_label.reshape(30, 1) * max(label) - model.predict(test_data) * max(label)
# 把誤差分成 15 等份, 求出每一等份的長度
step = (max(error) - min(error)) / 15
# 寫出每一等份的值
interval = [i for i in range(int(min(error)), int(max(error)) + int(step), int(step))]
# 實際預測比較圖
width = 0.3
plt.subplot(2, 1, 1)                  # 第一張圖位於視窗裡的位置 (2列1行的第二個位置 - 上)
plt.xlabel("test data")              # x軸名稱
plt.ylabel("money")
plt.bar(np.linspace(1, 30, 30) - width / 2, (test_label * max(label)).reshape(30), width =
width, label = 'actual')
plt.bar(np.linspace(1, 30, 30) + width / 2, (model.predict(test_data) *
max(label)).reshape(30), width = width, label = 'predict')
plt.legend()
```

# 最後測試：資料預測

```
# 建立欲預測的資料
```

```
data = np.array([[8, 5, 0, 0],  
                [15, 6, 0, 0],  
                [12, 5, 1, 0],  
                [17, 2, 1, 0]])
```

```
# 資料正規化與預測資料
```

```
data = data - mean          # data 減掉平均數  
data = data/std            # data 除以標準差  
tem = model.predict(data)  # 得出預測值  
tem = tem * max(label)    # 還原標籤資料  
print(tem)                # 顯示標籤資料
```

# Homework

回歸分析：學生身高體重分析

(請省去建立資料比較圖，只查看損失值即可)

繳交 Word 檔：

1. 原始數據資料，與你的程式碼
2. loss 與 val\_loss 損失值比較圖
3. 預測自訂資料的結果
4. Email to : [phd9512@cs.nchu.edu.tw](mailto:phd9512@cs.nchu.edu.tw)

# 迴歸問題

某一班學生的身高和體重是否相關?

能否用身高來推測某位學生的體重?

| 身高  | 體重 |
|-----|----|
| 150 | 40 |
| 152 | 48 |
| 155 | 45 |
| 158 | 50 |
| 160 | 55 |
| 162 | 56 |
| 165 | 58 |
| 170 | 59 |
| 172 | 62 |
| 175 | 65 |
| 180 | 68 |
| 185 | 72 |