



用Python學智慧聯網

Chapter 02 : 用 Python 玩轉 AI

踏入 AIoT 的世界

旗標創客



↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 1 何謂 AIoT

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 2 用Python 玩轉 AI

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 3 AI 的小大腦 – 微控制器

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 4 迴歸問題 – 體溫監測站

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 5 IoT 應用 – 體溫通報器

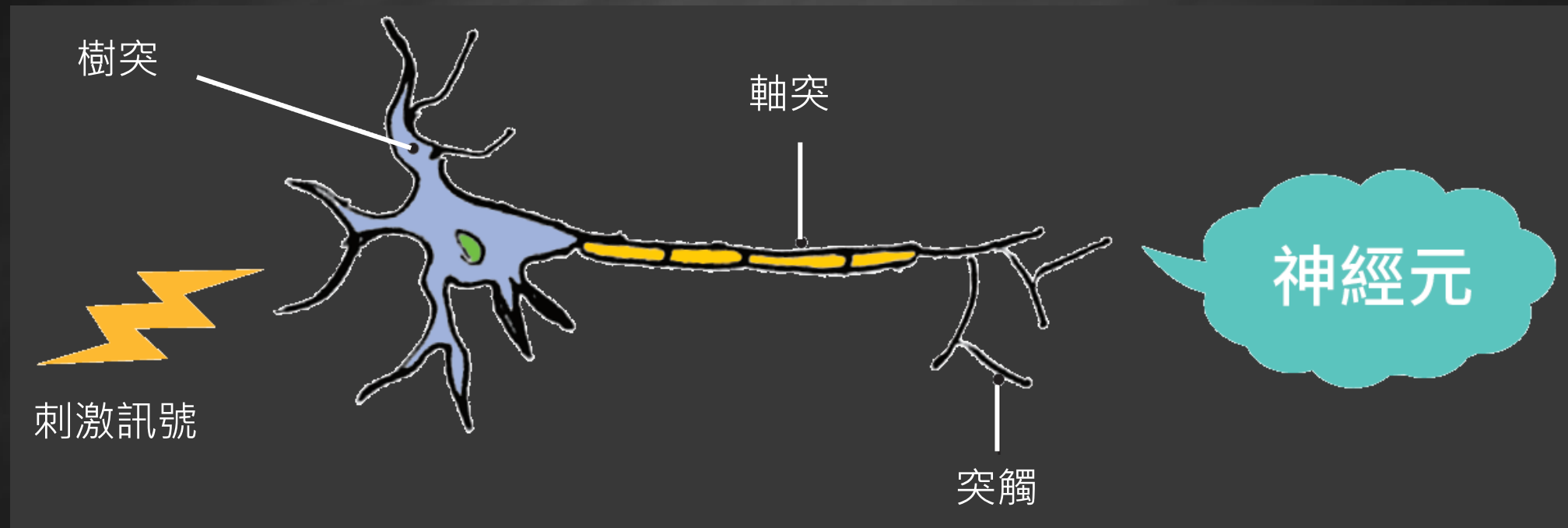
↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 6 二元分類 – 雲端步頻紀錄儀

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 7 多元分類 – 無線體感鍵盤

↑ ↓ ↻ 消息 設定 垃圾桶 ⋮
▶ 8 CNN – 智慧聲控燈

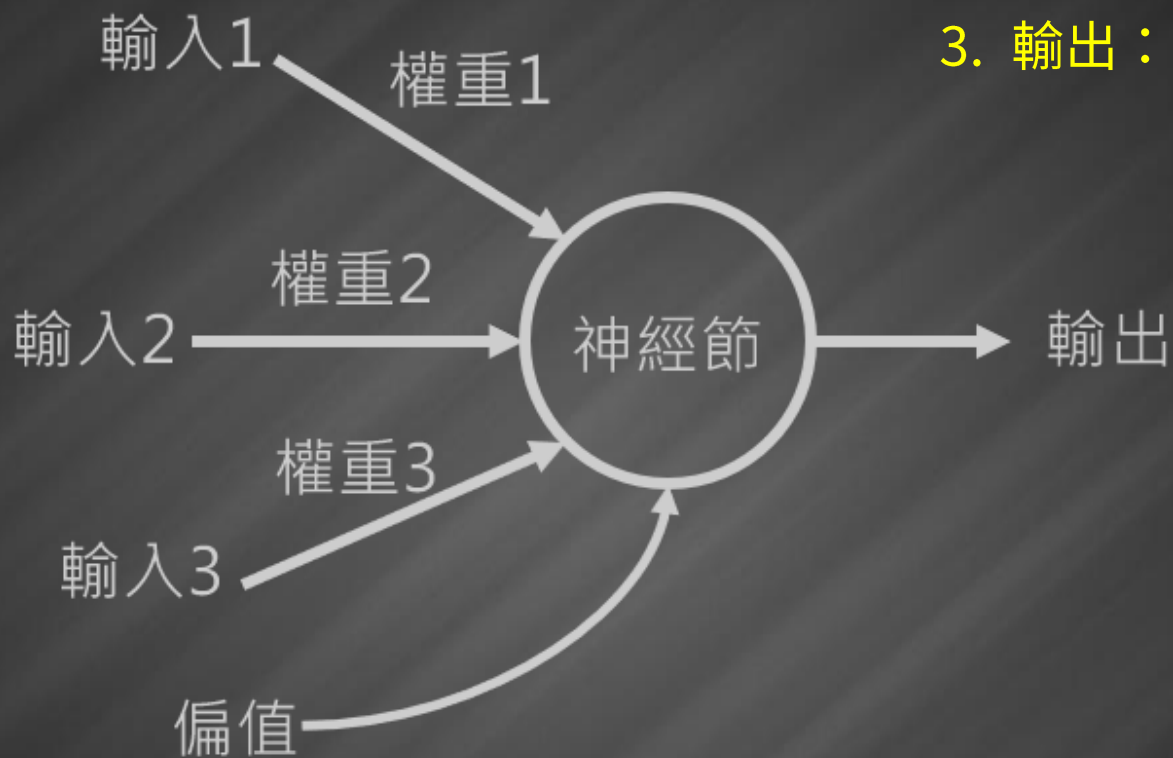
旗標創客

初探 AI-神經網路 (主流的機器學習技術)



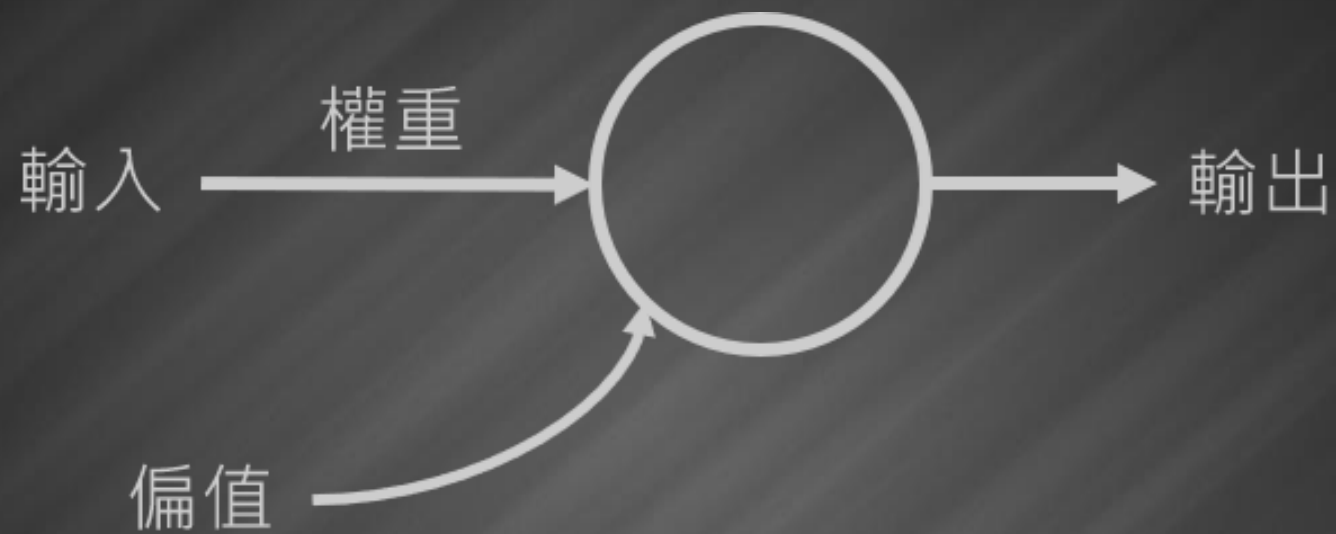
人工神經元

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

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

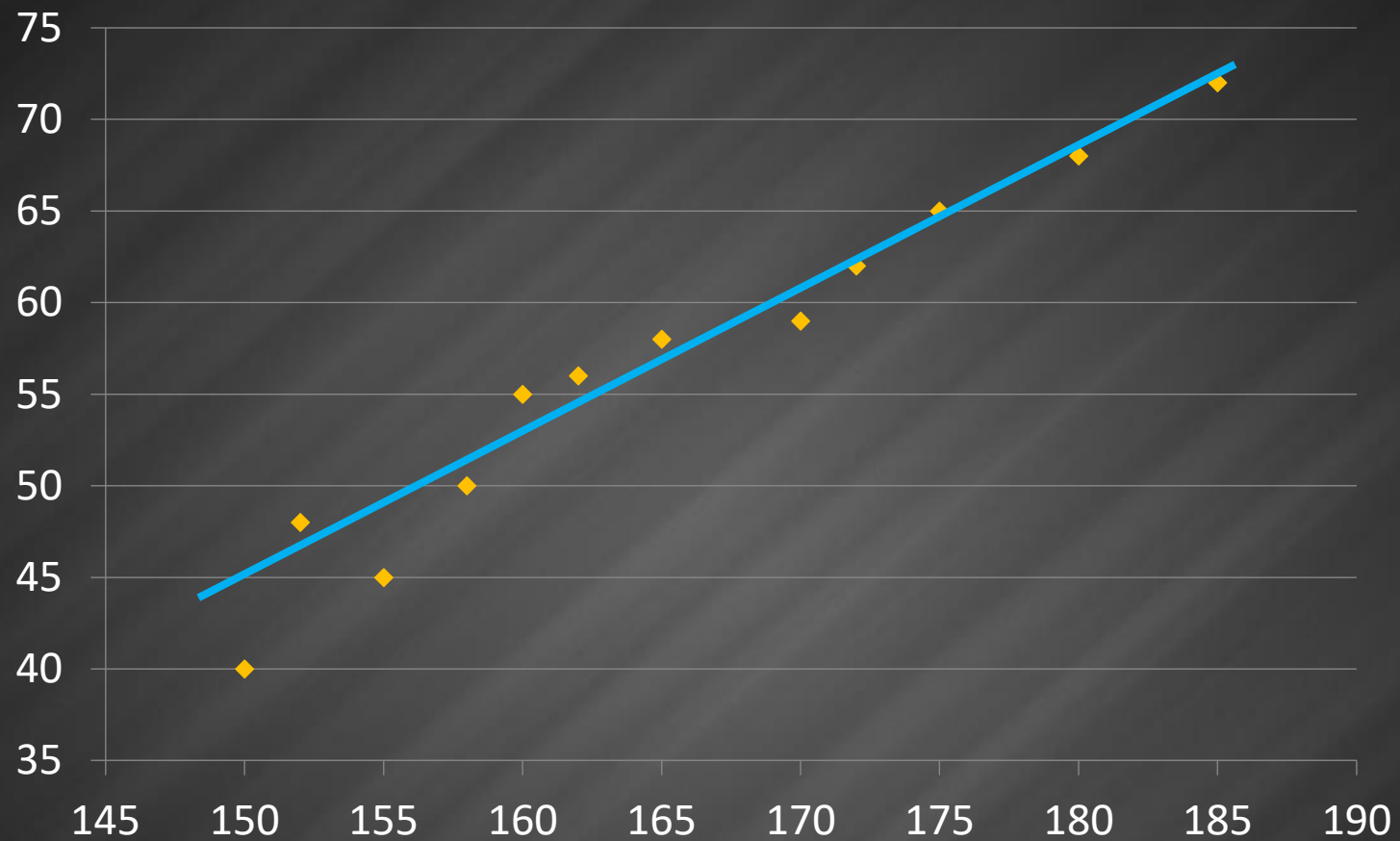
某一班學生的身高和體重是否相關?

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)

輸出 = 輸入 × 權重 + 偏值



$$y = 0.8337x - 81.331$$

迴歸線 (函數)

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



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

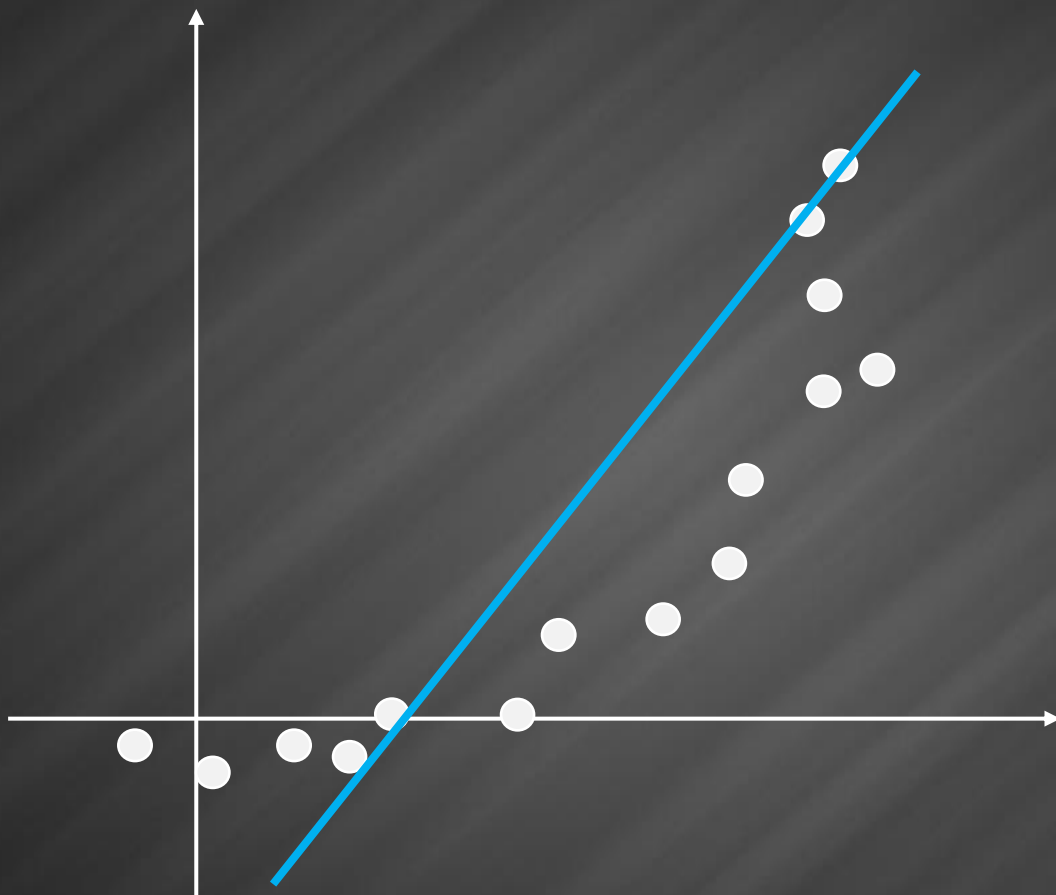
建立兩組資料間的對應函數

非線性問題

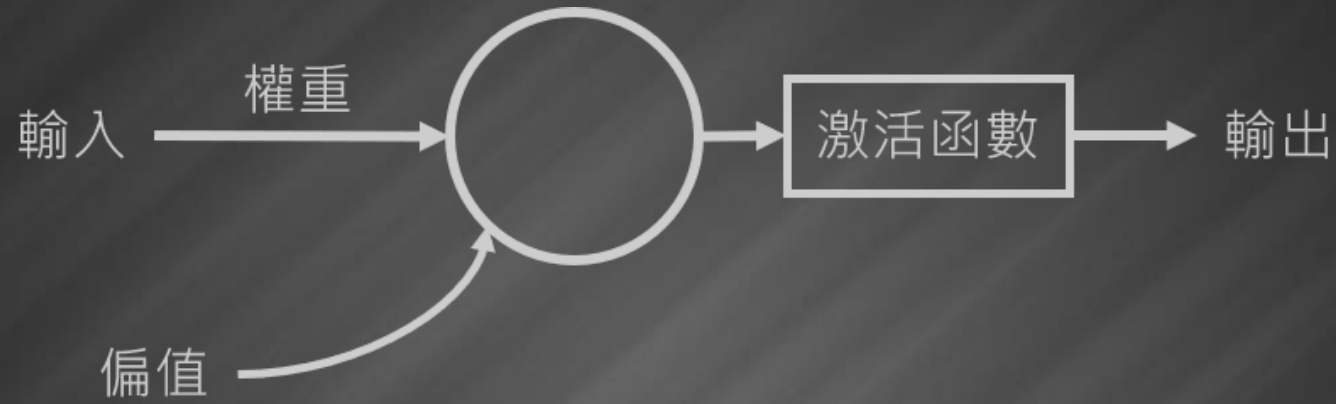
輸出 = 輸入 × 權重 + 偏值

線性函數 (又稱一次函數)

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

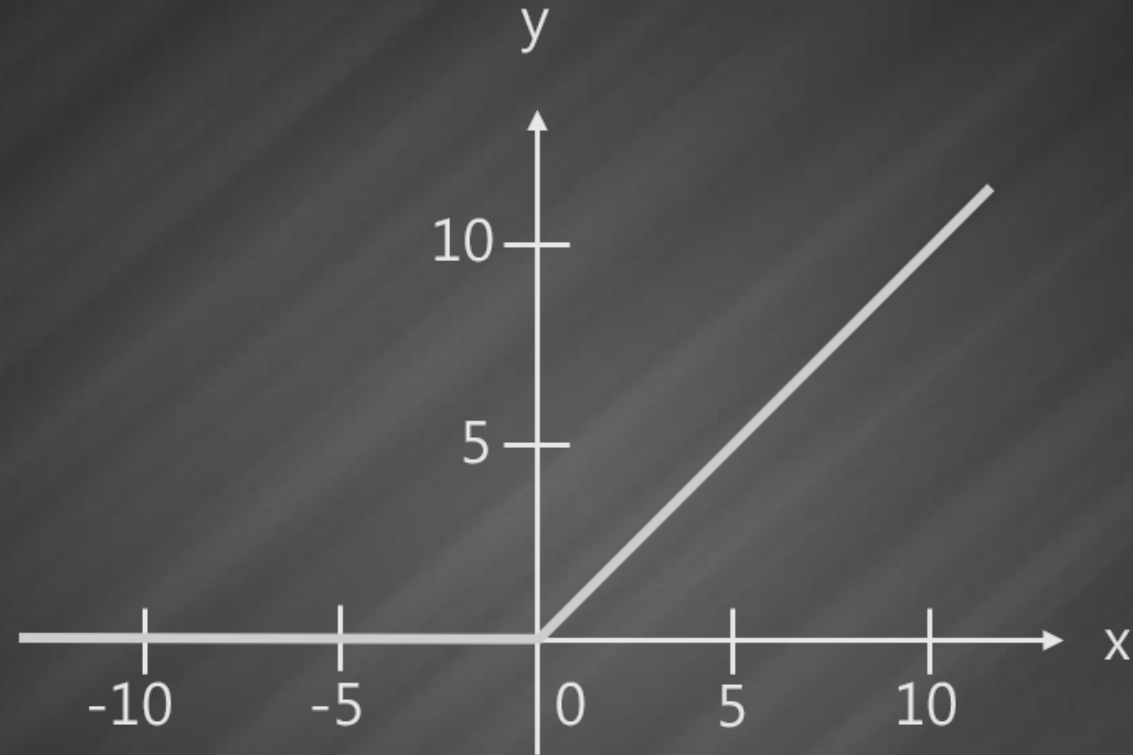


激活函數 (activation function)



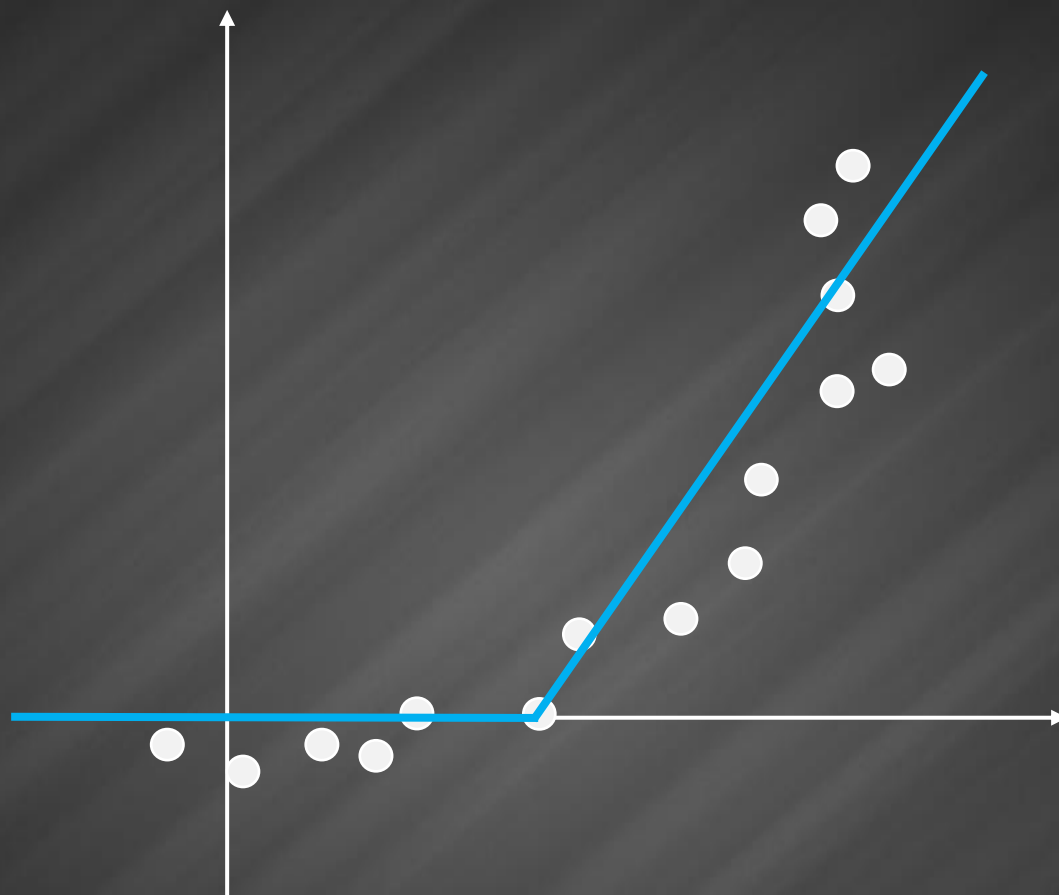
$$\text{輸出} = \text{激活函數}(\text{輸入} \times \text{權重} + \text{偏值})$$

ReLU 函數 (線性整流函數, 增加非線性度)



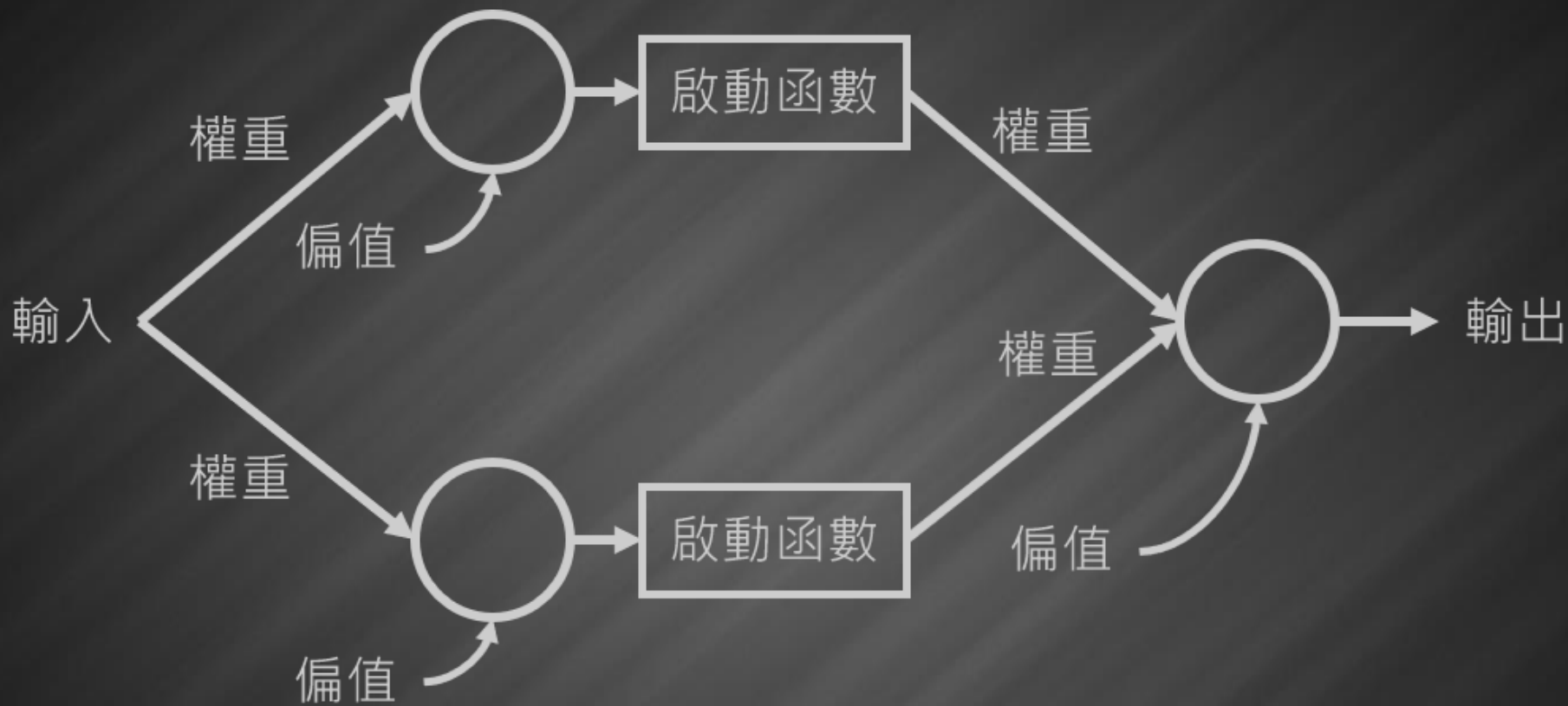
小於 0 就等於 0

ReLU 函數 (線性整流函數, 增加非線性度)

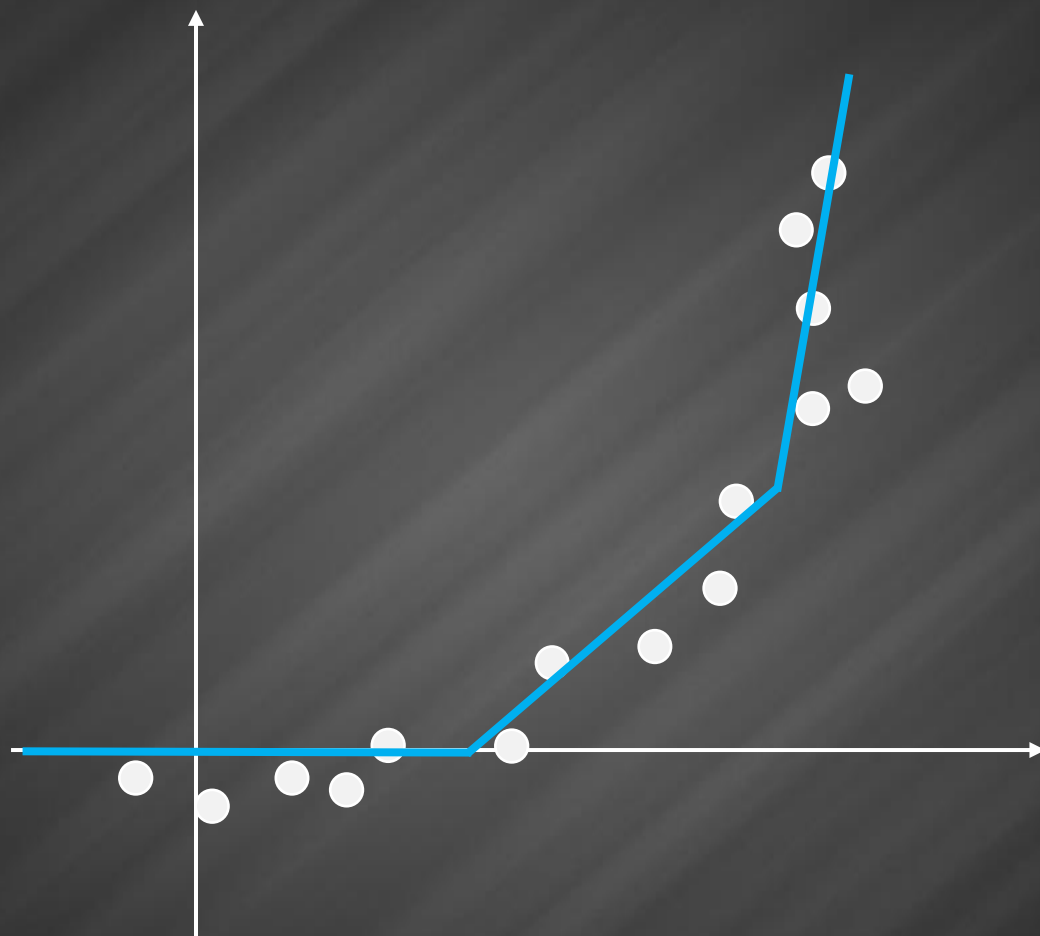


小於 0 就等於 0

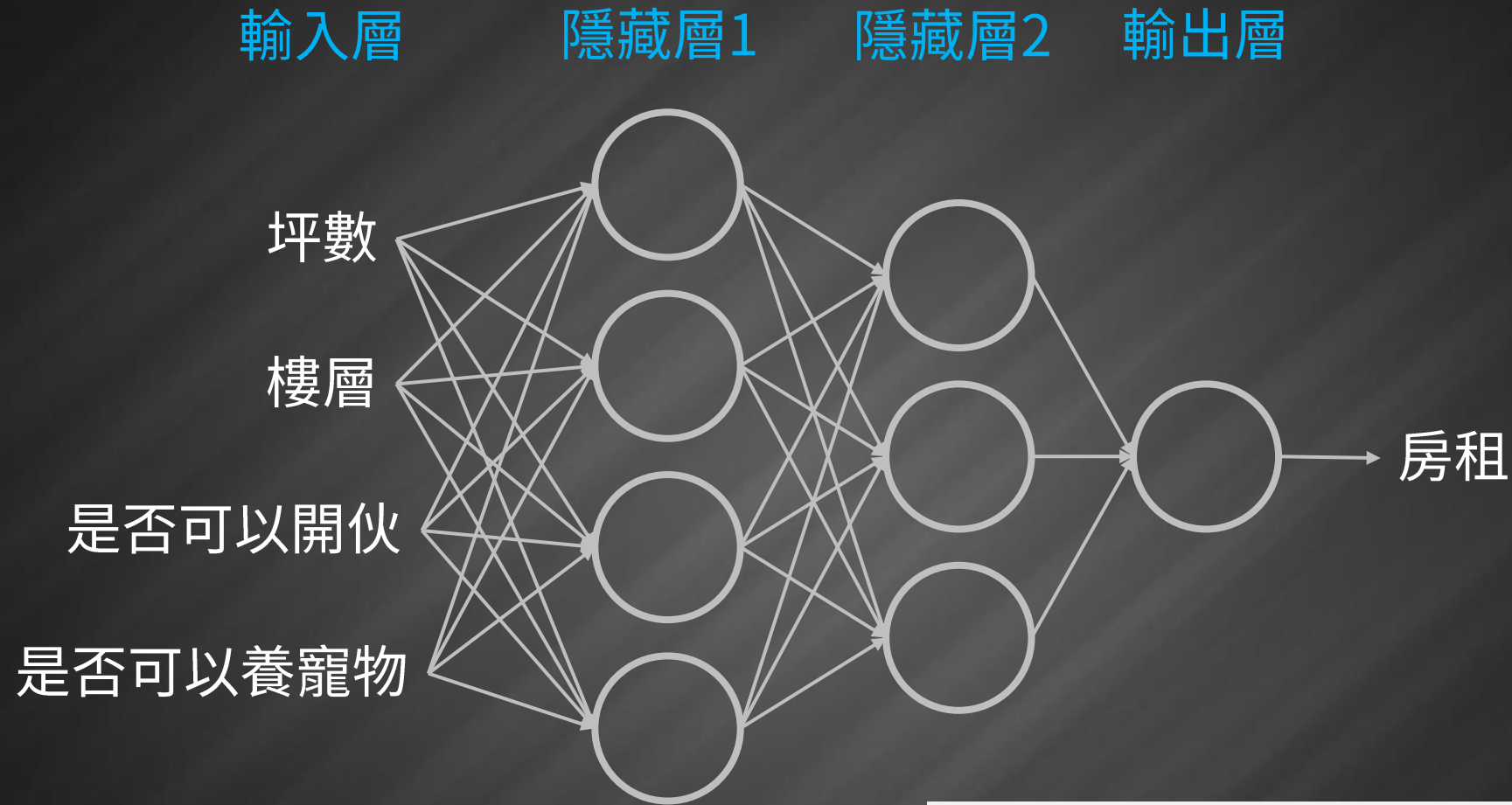
更貼近資料：多神經元串聯



更貼近資料：多神經元串聯

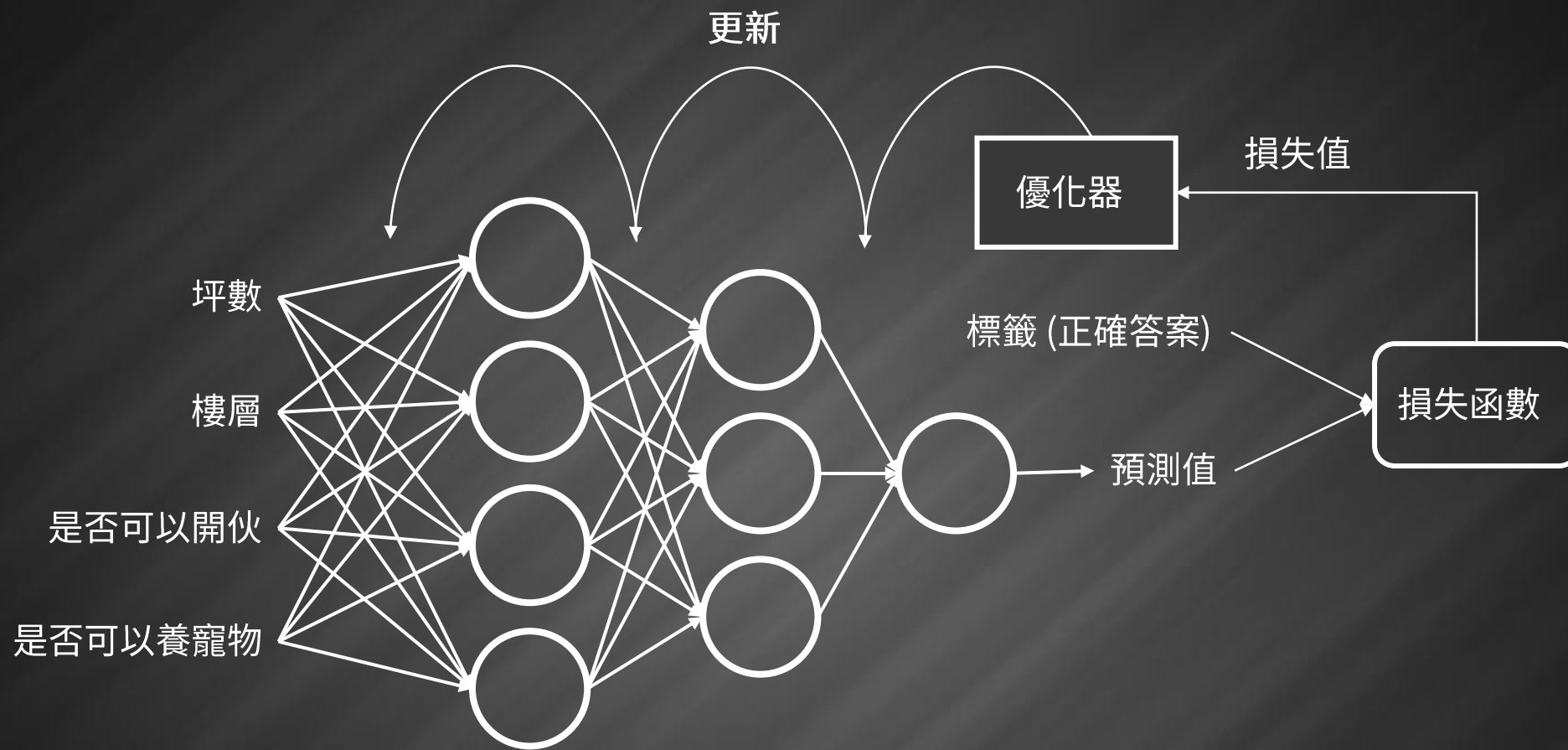


神經網路 (又稱為模型)



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：

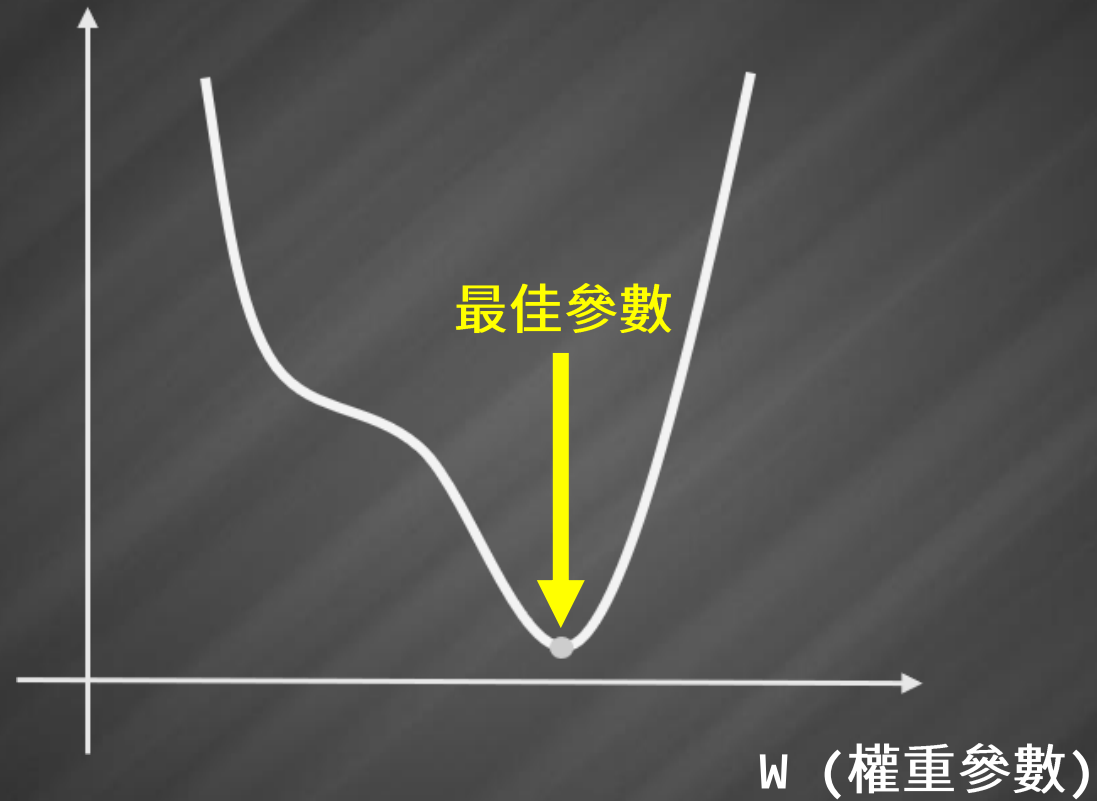
$$\hat{y}_1、\hat{y}_2、\hat{y}_3、\hat{y}_4、\hat{y}_5 \cdots \hat{y}_n$$

MSE：

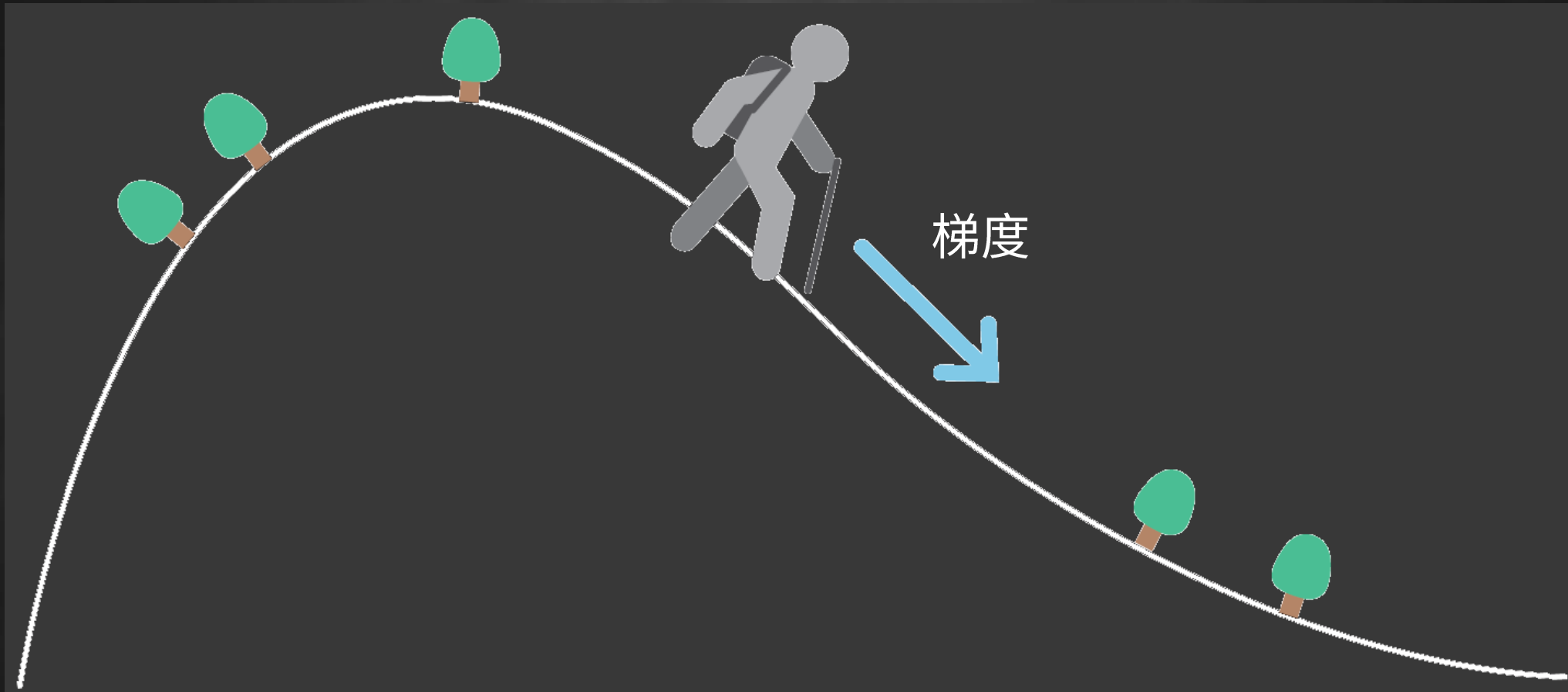
$$\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

優化器：使用梯度下降 (Gradient descent)

loss (損失值)

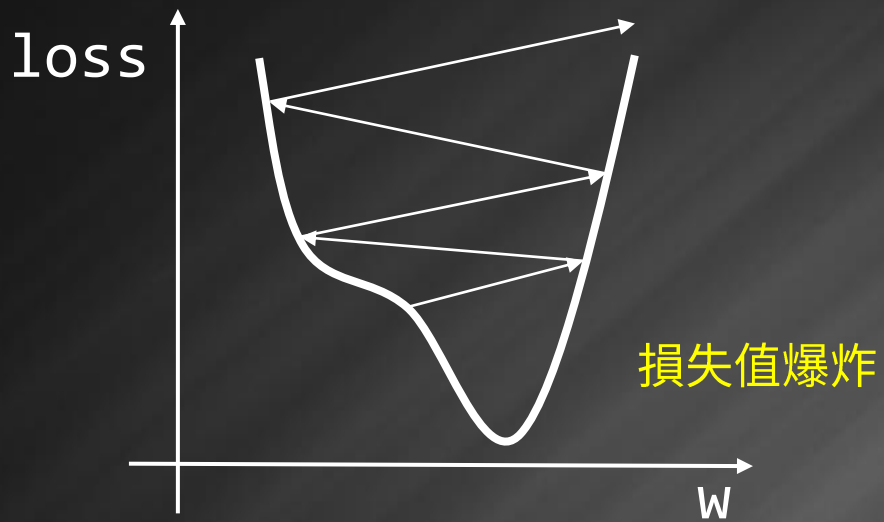


優化器：使用梯度下降 (Gradient descent)

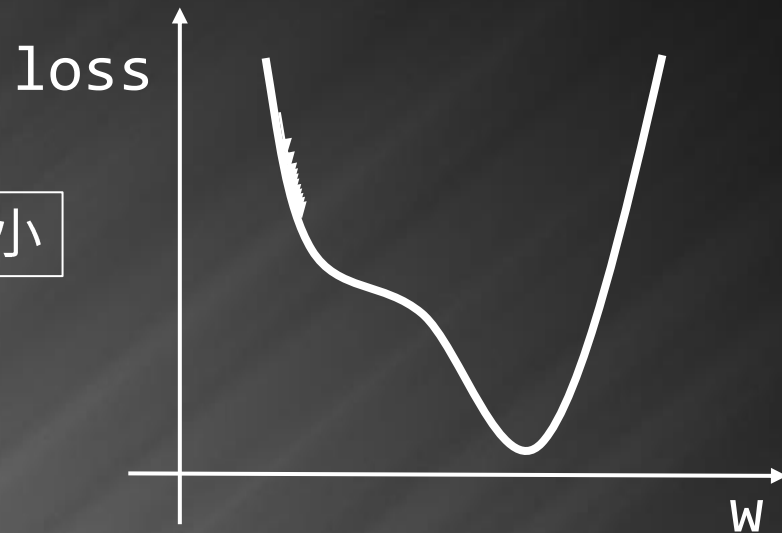


學習率：介於 $0 \sim 1$ (調整步伐大小)

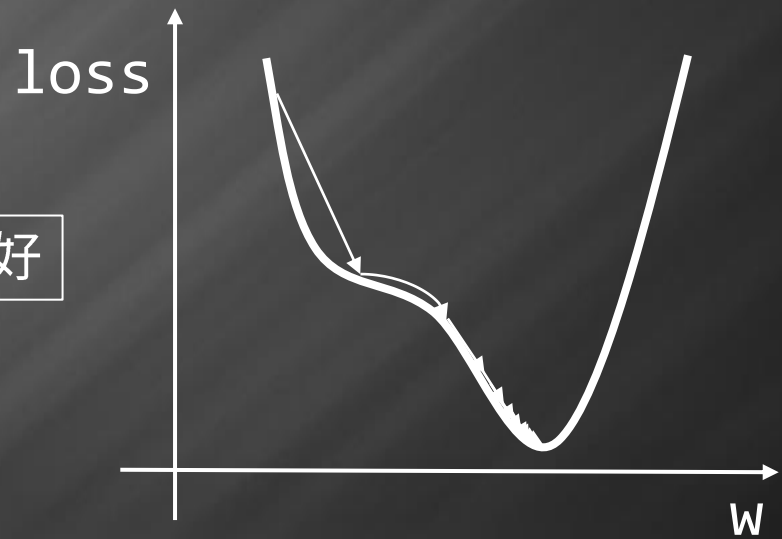
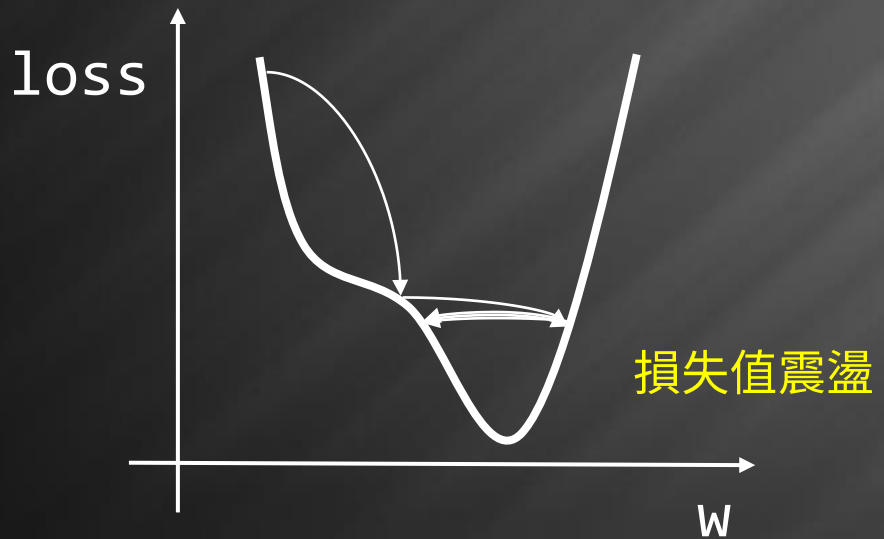
太大



太小

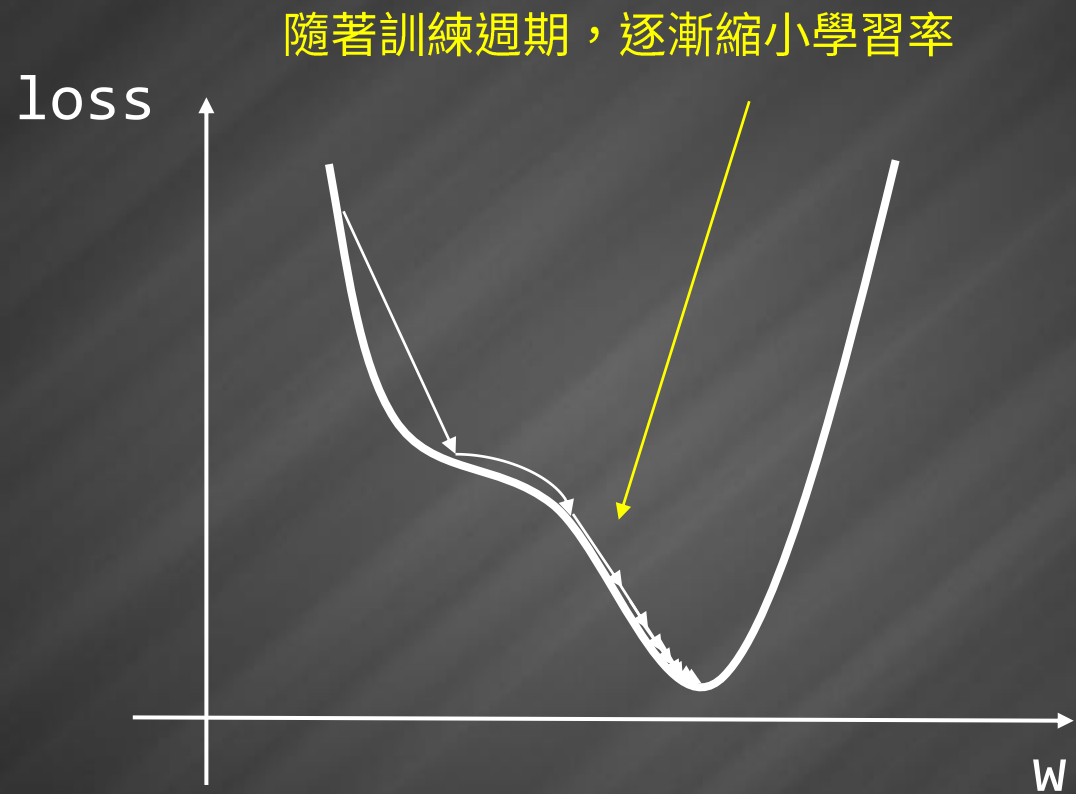


剛好



自適應 (Adaptive)

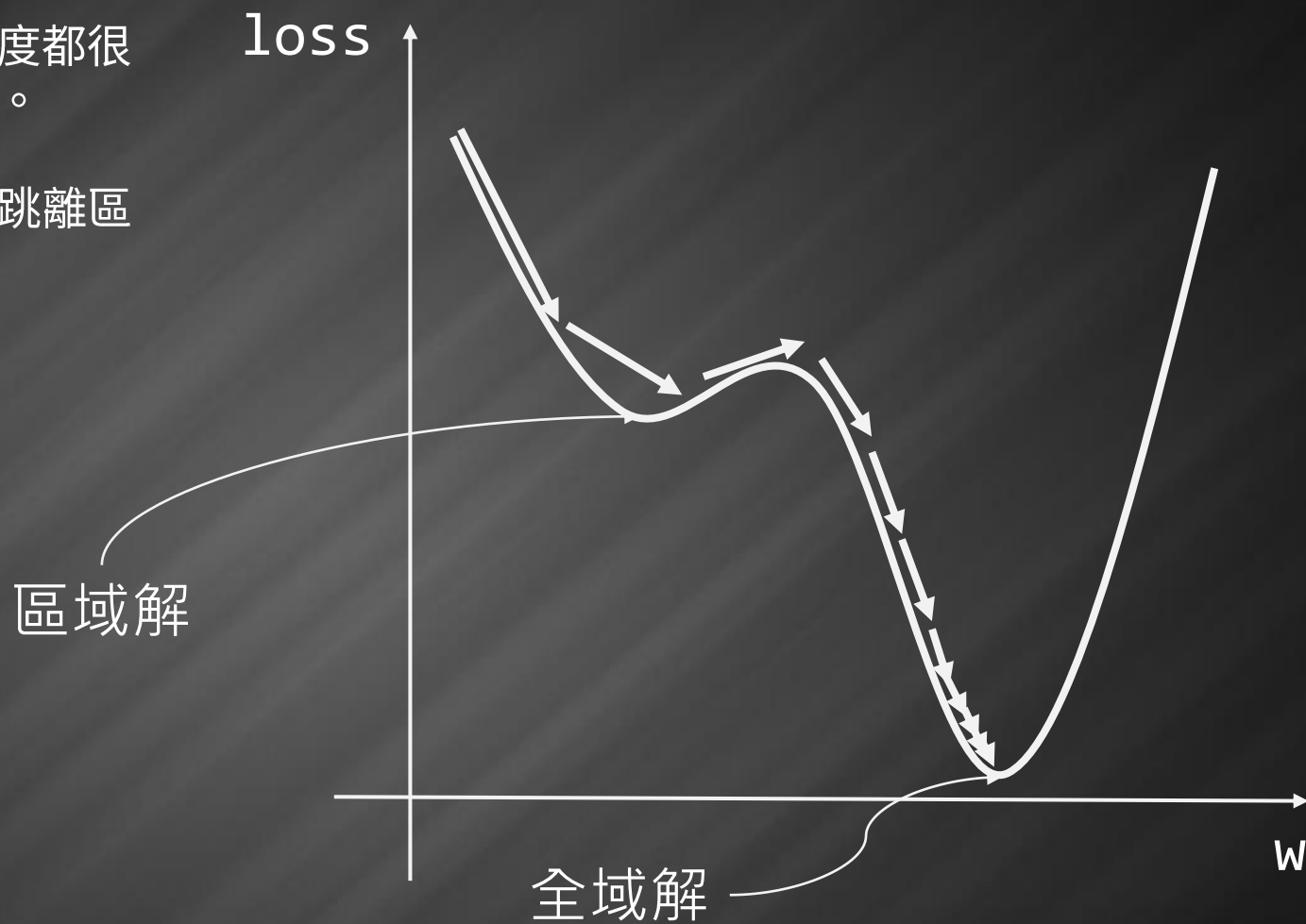
- 自適應：自動調整學習率



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。



學習 AI 時的重要工具 - Python





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 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

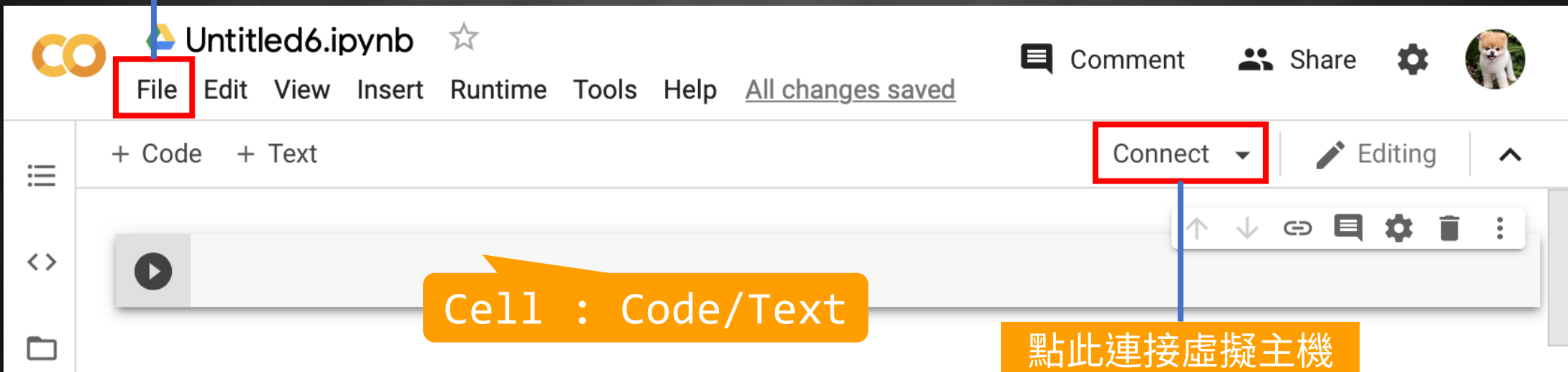
The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

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
```

CO 快速上手 - 新建筆記本

File/New notebook



Cell : Code/Text

點此連接虛擬主機



快速上手 - 執程式碼

The screenshot shows a Jupyter Notebook interface with several annotations:

- 改檔名** (Rename): Points to the filename `Untitled6.ipynb` in the top header.
- 刪除/複製/剪下Cell** (Delete/Clone/Cut Cell): Points to the cell action menu (up, down, link, comment, settings, trash, and more options) located below the RAM/Disk status bar.
- 執行程式碼** (Execute Code): Points to the play button icon on the left side of the code cell.
- 滑鼠移到此處可增加 Cell** (Move mouse here to add Cell): Points to the `+ Code` and `+ Text` buttons located below the code cell.

The notebook content shows a code cell with `print('Hello Python!')` and its output, `Hello Python!`.

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, and the 'Change runtime type' option is highlighted with a red box and a yellow '2' label. A yellow '1' label is also present near the 'Runtime' menu header. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus. The 'Runtime' menu contains options such as 'Run all', 'Run before', 'Run the focused cell', 'Run selection', 'Run after', 'Interrupt execution', 'Restart runtime...', 'Restart and run all...', and 'Factory reset runtime'. The 'Change runtime type' option is at the bottom of the menu. The interface also shows a 'RAM' and 'Disk' usage indicator, a 'Comment' button, a 'Share' button, and a user profile picture.

CO Untitled6.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

print('Hello Python')

Hello Python!

RAM Disk

Editing

↑ ↓ ↻ 🗨 ⚙ 🗑 ⋮

1

2

Change runtime type



雲端主機的管理與設定 - GPU 加速

The screenshot shows the 'Notebook settings' dialog box in a Jupyter Notebook environment. The dialog is titled 'Notebook settings' and has a white background. It contains the following elements:

- Runtime type:** A dropdown menu currently set to 'Python 3'.
- Hardware accelerator:** A dropdown menu currently set to 'None'. A red box highlights the dropdown arrow (3), and a red box highlights the 'GPU' option in the dropdown menu (4).
- Omit code cell output:** A checkbox that is currently unchecked.
- Buttons:** 'CANCEL' and 'SAVE' buttons at the bottom right. A red box highlights the 'SAVE' button (5).

Numbered callouts (3, 4, 5) are placed on orange boxes to indicate the steps for selecting GPU as the hardware accelerator.




雲端主機的管理與設定 - Session 管理

The screenshot displays the CO IDE interface for a file named 'Untitled6.ipynb'. The 'Runtime' menu is open, showing various execution options. A red box highlights the 'Runtime' menu item, and a yellow box with the number '1' is placed next to it. Another red box highlights the 'Manage sessions' option at the bottom of the menu, with a yellow box containing the number '2' next to it. In the top right corner, there is a yellow button labeled '查看當前資源用量' (View current resource usage). Below this button, a red box highlights the 'RAM' and 'Disk' resource usage indicators, which are shown as progress bars. The main editor area contains a code cell with the text `print('Hello Python')` and its output, 'Hello Python!'. The interface also shows a sidebar with '+ Code' and '+ Text' options, and a top navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', and 'Help' menus.



雲端主機的管理與設定 - Session 管理

Active sessions

Title	Last execution	RAM used	
 Untitled6.ipynb Current session	0 minutes ago	0.15 GB	TERMINATE

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 shows the JupyterLab interface for a file named 'Untitled6.ipynb'. The top menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status indicator 'All changes saved'. On the right, there are 'Comment', 'Share', and a settings gear icon. The 'Files' sidebar on the left contains an 'Upload' button (highlighted with a red box), a 'Refresh' button, and a 'Mount Drive' button (highlighted with a red box). Below these is a folder icon (highlighted with a red box) and a folder named 'sample_data'. At the bottom of the sidebar, a 'Disk' progress bar shows '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 interface also features '+ Code' and '+ Text' buttons, 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 shows the Jupyter Notebook interface for a file named 'Untitled6.ipynb'. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help', with a status indicator 'All changes saved'. On the right, there are buttons for 'Comment', 'Share', and a user profile picture.

The left sidebar, titled 'Files', contains an 'Upload' button, a 'Refresh' button, and a 'Mount Drive' button. The 'Mount Drive' button is highlighted with a red rectangular box, and a yellow square with the number '1' is placed to its right. Below these buttons are folder icons for '..' and '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 interface of a Jupyter Notebook. The title bar reads "Untitled6.ipynb" with a star icon. Below it is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", followed by the text "All changes saved". On the right side of the top bar are icons for "Comment", "Share", a settings gear, and a user profile picture. Below the menu bar is a toolbar with "+ Code", "+ Text", a RAM/Disk usage indicator, and an "Editing" mode selector. On the left, a "Files" sidebar is visible with options for "Upload", "Refresh", and "Mount Drive".

A white modal dialog box is centered on the screen with the following text:

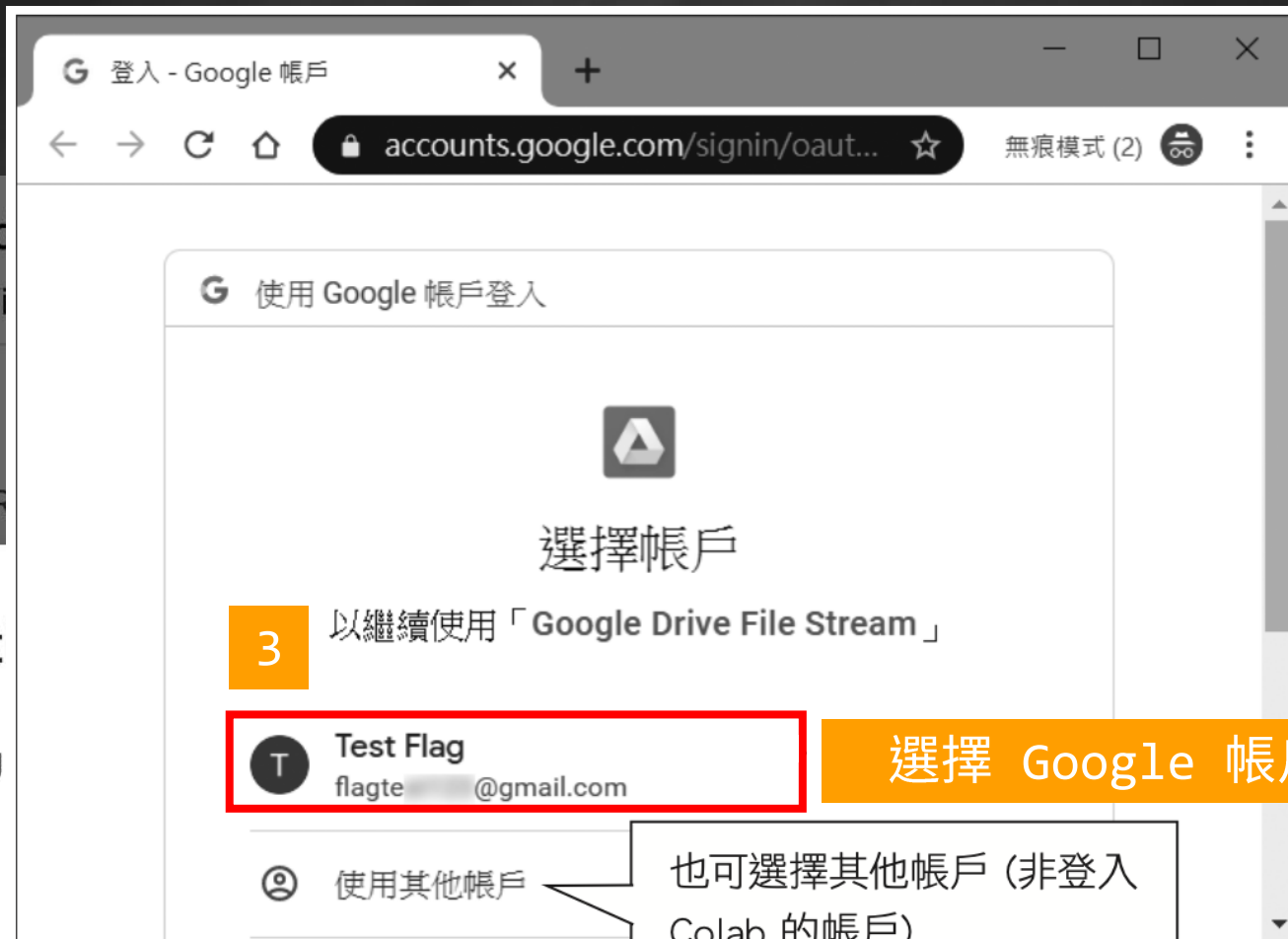
Permit this notebook to access your Google Drive files?

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

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 rectangular border. A small orange square with the number "2" is positioned above the "CONNECT TO GOOGLE DRIVE" button.



目錄窗格與檔案管理 – 掛載雲端



CO Untitled

File Edit View

Files

Upload Refresh

Permissions

Connecting

選擇 Google 帳戶

也可選擇其他帳戶 (非登入 Colab 的帳戶)

GOOGLE DRIVE



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

Files × + Code + Text RAM Disk Editing ^

Upload Refresh Unmount Drive

drive 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

ex2-3

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/
```
- Annotations:** Four orange boxes with white text provide instructions:
 - 1. 按右鍵取得路徑 (Click right button to get 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 (Decompression 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  

Command palette

Settings...

Keyboard shortcuts... ⌘/Ctrl+M H !')

Files

Upload Refresh Unmount

..

20191104_labelme

drive

sample_data

20191104_labelme.zip

Hello Python!

!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 a code editor. The 'Site' category is selected in the left sidebar. The 'Theme' dropdown menu is open, showing three options: 'dark', 'light', and 'adaptive'. The 'dark' option is highlighted with a red border and a yellow '2' in a square, 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 visible. At the bottom of the dialog, there are 'CANCEL' and 'SAVE' buttons. The background shows a file explorer on the left and a status bar at the bottom indicating 'Disk 79.38 GB available'.

Settings

Site

Editor

Colab Pro

Miscellaneous

Theme

dark light

dark 2

adaptive

New notebooks use...

Request GitHub authentication for private repositories and organizations

[More info](#)

Custom snippet notebook URL

CANCEL SAVE

Disk 79.38 GB available



偏好設定 - 柯基犬與小貓模式

The screenshot shows the 'Settings' dialog box for the CO application. The 'Miscellaneous' category is selected in the left sidebar. The main area shows the 'Editor' section with 'Corgi mode' and 'Kitty mode' checked. The 'SAVE' button is highlighted at the bottom right.

Category	Item	Status	Count
Editor	Corgi mode	Checked	4
	Kitty mode	Checked	4
Miscellaneous	(Selected)	-	3
Buttons	SAVE	Highlighted	5



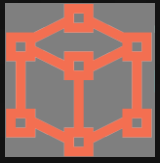
偏好設定 - 柯基犬與小貓模式

The screenshot displays the JupyterLab interface with the following elements:

- Top Bar:** CO logo, file name "Untitled6.ipynb", and a star icon. The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status indicator "All changes saved".
- File Browser (Left):** Shows a file tree with folders "20191104_labelme", "drive", and "sample_data", and a file "20191104_labelme.zip".
- Code Editor (Center):** Contains a code cell with the following code:

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

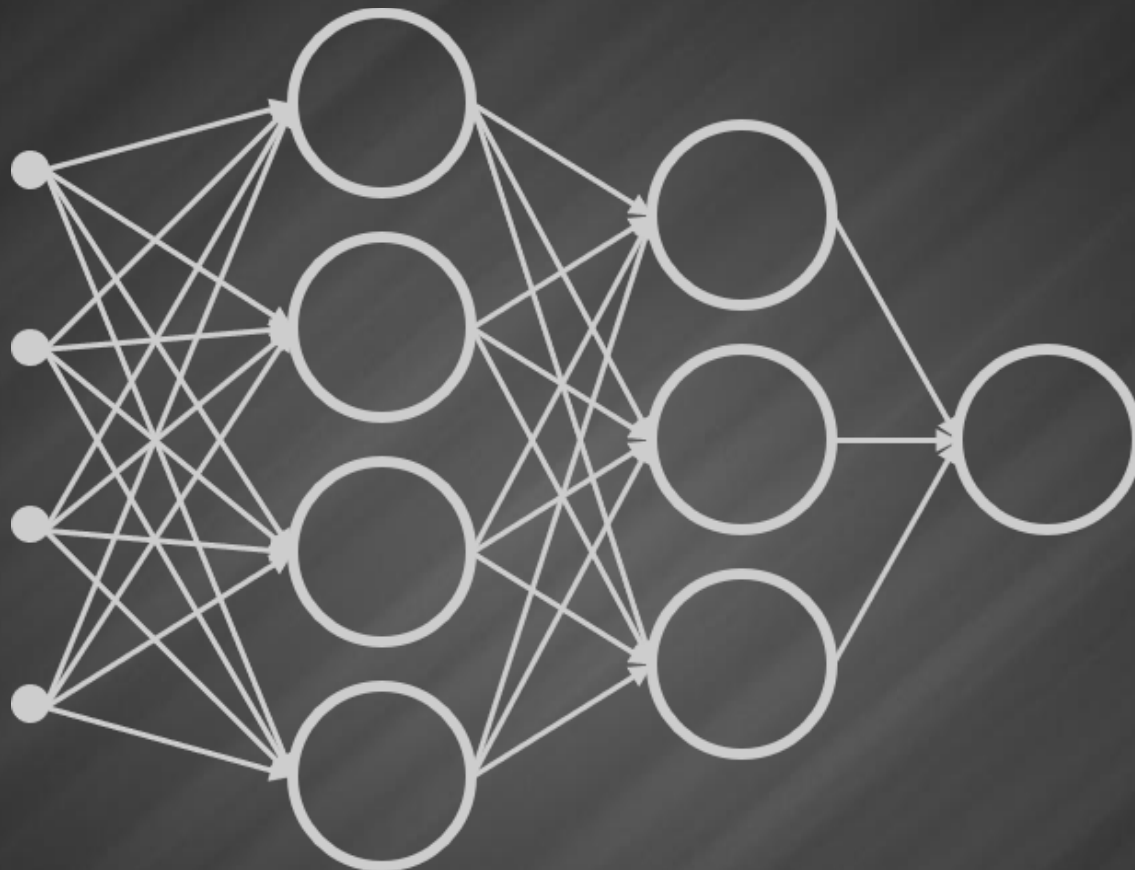
Below the code, the output is displayed as "Hello Python!".
- Terminal (Bottom):** Shows a terminal prompt with the command: `!unzip /content/20191104_labelme.zip`
- Right Panel:** Includes RAM and Disk usage indicators, a "Editing" mode selector, and a "Share" button.
- Decorations:** A corgi and a cat are visible in the top right area of the interface.
- Bottom Status Bar:** Shows "Disk" usage and "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

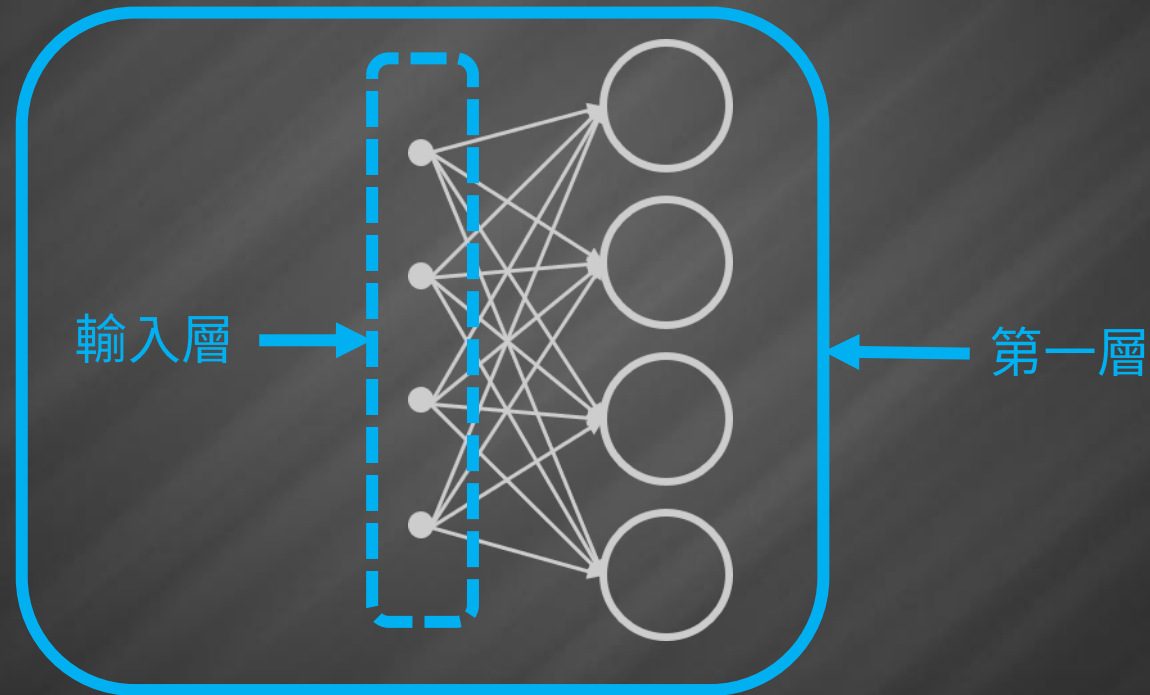
ex2-4

```
# 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）

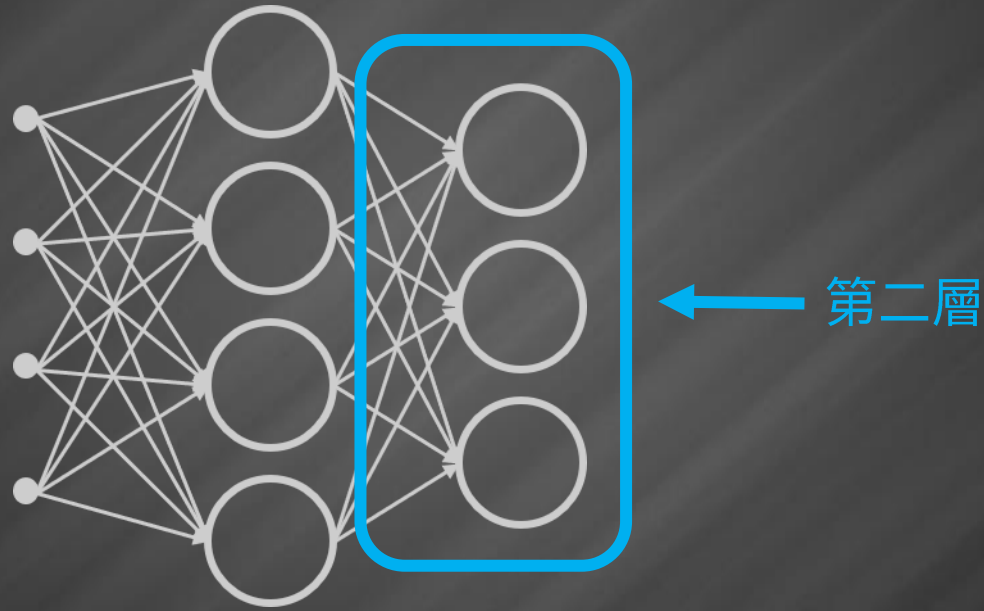
```
model.add(Dense(4, activation='relu', input_shape= (4, ))) ← 輸入層形狀
```

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

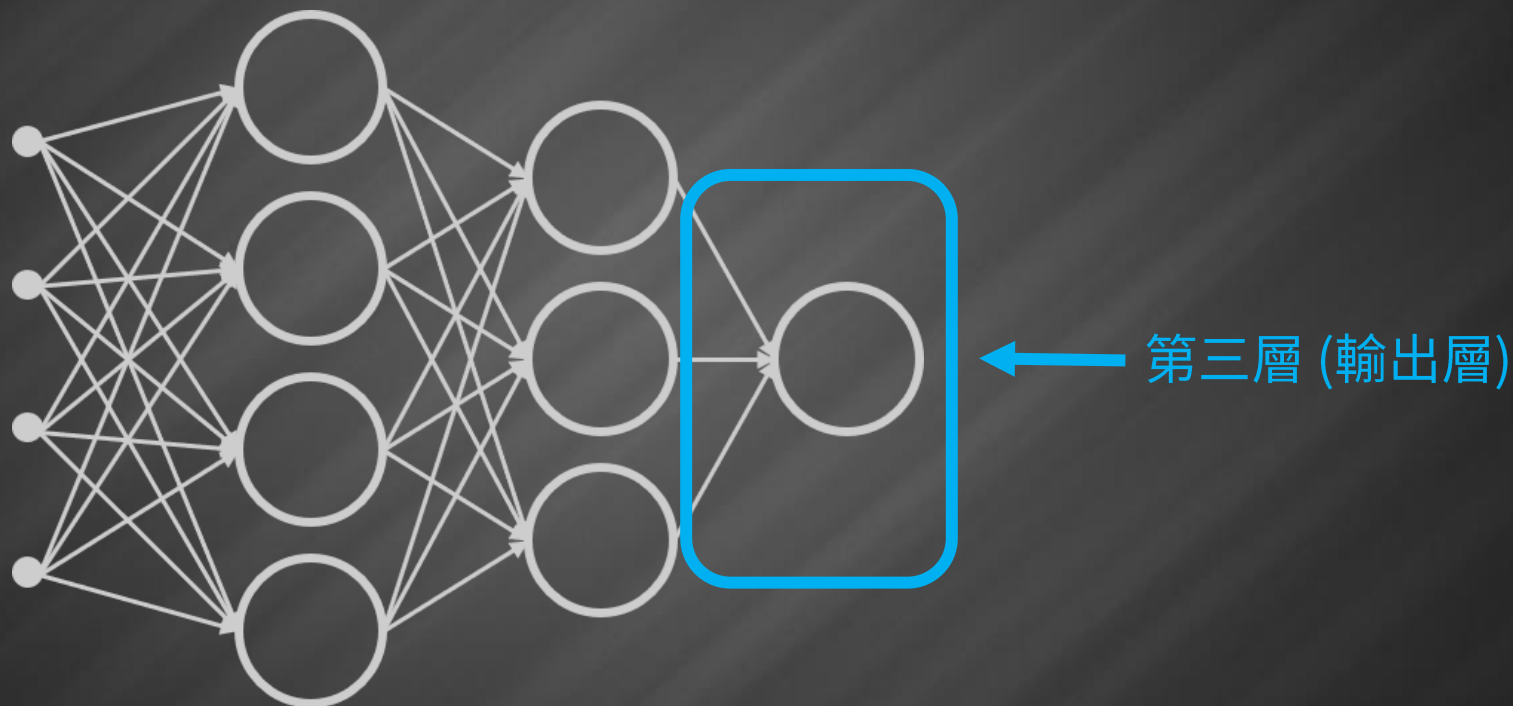
```
model.add(Dense(3, activation='relu')) ← 除第一層之外都不用指定  
input_shape 參數
```



加入輸出層

ex2-4

`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

ex2-4

```
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：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 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.array 格式)
```

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

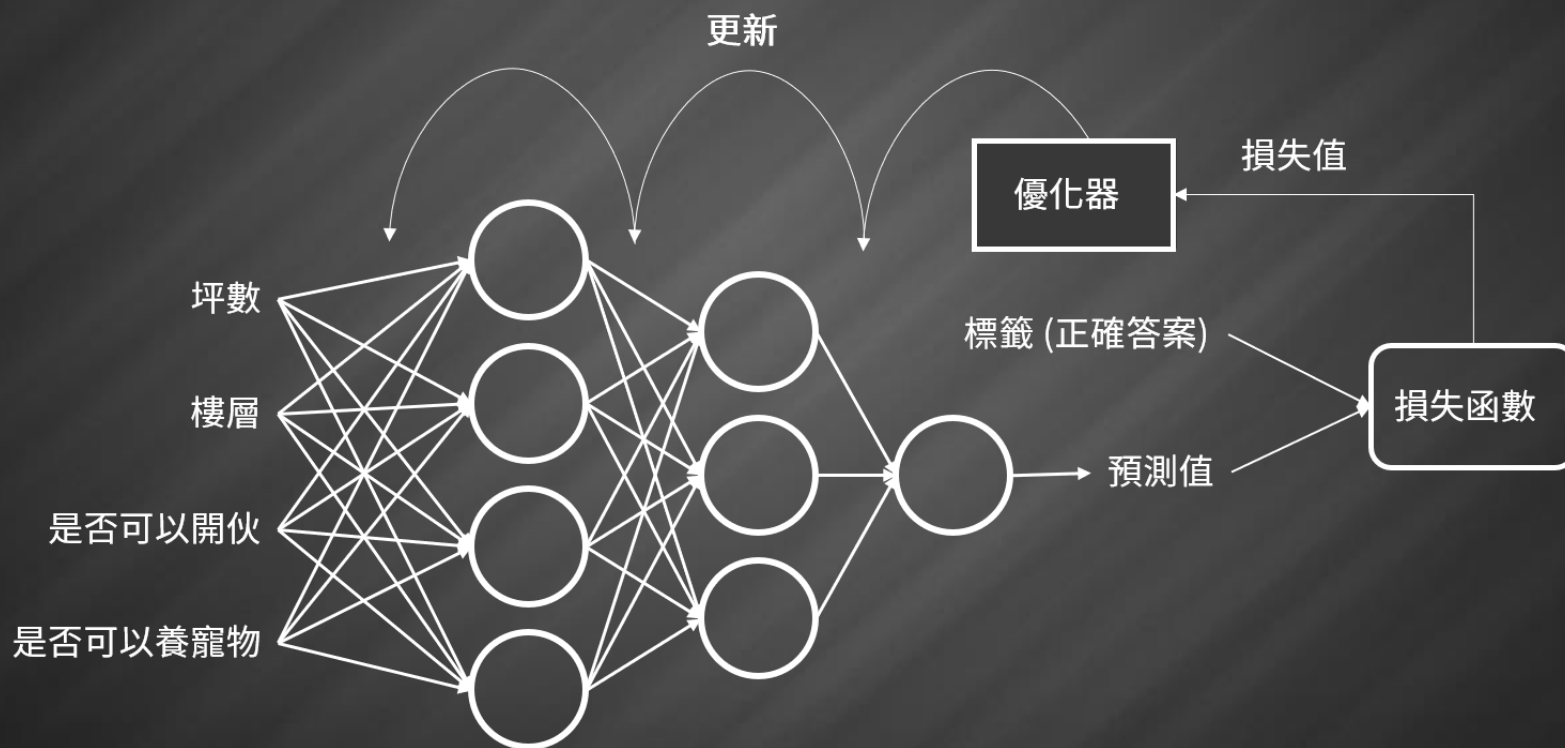
>>> len(a)                        # len() 會回傳容器內元素的數量

>>> 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]
```

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： $x_1 = 5, x_2 = 6, x_3 = 8, x_4 = 9$

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

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

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

$\bar{x} = 7$ 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
```

標籤正規化

ex2-9

```
# 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)。
 - ✓ 測試集：10% 中的最後 30 筆。
 - ✓ 驗證集：剩下的當驗證集 (39 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
print(train_data.shape)
train_label = new_label[:split_num] # 訓練用標籤
# 驗證集
validation_data = data[split_num:-30] # 驗證用資料
print(validation_data.shape)
validation_label = new_label[split_num:-30] # 驗證用標籤
# 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
```

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) # 增加輸出為 1 的輸出層
```

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
```

```
# 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()
```

```
# 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
```