CHAPTER 1

人工智慧簡介

本章先從關於人工智慧（Artificial Intelligence, AI）的簡短介紹開始，這有助於提供一個完整的框架，讓我們理解 AI 究竟是什麼，以及它如何成為一個引人入勝又快速發展的研究領域，就從關於 AI 起源的歷史事件開始。

AI 的歷史起源

AI，或是類似的東西，顯然已經存在很長一段歲月了。根據文獻，古希臘哲學家就曾討論到自動裝置（automaton）或所謂本身具有一定智慧的機器。西元 1517，來自布拉格的 Golem 誕生了；如圖 1-1。

類型	簡述
電腦視覺 Computer vision	電腦能夠由數位影像 / 影片中獲得高階資訊的跨領域學科。
演化運算 Evolutionary computing	演化演算法是以達爾文的演化論為基礎也因此得名。 這些演算法屬於試誤型問題解決器的一種，並運用 啟發式或隨機式的總體性方法來判斷各種解法。
遊戲 AI Gaming AI	將 AI 用於遊戲來產生智慧行為，主要是用於非人類角色（NPC），通常是模擬類人智慧。
人機介面（HCI）	HCI 研究電腦科技的設計與運用方式，專注於人（使用者）與電腦之間的各種介面。
智慧軟體助理或智慧個人助理 Intelligent soft assistant or intelligent personal assistant (IPA)	代理軟體（software agent）可為我們執行各種任務或服務。這些任務或服務通常是根據使用者輸入內容、所在地，以及從各種線上資源獲得資訊的能力而定。 這類軟體包括 Apple Siri、Amazon Alexa、Amazon Evi、Google Home、Microsoft Cortana、開放源碼的 Lucida、Braina（由 Brainasoft 針對 Microsoft Windows 作業系統所開發）、Samsung S Voice 以及 LG G3 Voice Mate。
知識工程	關於製作、維護與使用知識系統所需的所有科技、科學與社會觀點
知識呈現 Knowledge representation（KR）	以電腦系統可用於解決複雜任務的形式來呈現關於這個世界的資訊，例如醫學診斷或以自然語言來對話。
邏輯程式設計	大幅仰賴正式邏輯的一種程式類型。任何用邏輯程式語言編寫的程式實際上是一組邏輯性的語句，用於呈現特定問題領域的事實與規則。主要的邏輯程式語言系列包含 Prolog、答案集程式（answer set program, ASP）與 Datalog。

續下頁

很明顯地，玩家端的最佳策略需要根據對手的上一步而定，這也是根據正常人思維來設計的。不過，要寫個檢查機制並不難，好避免給人類對手這樣的可趁之機。程式會長這樣；但請注意我用了整數值而非對應的字串。

```
if computer == lastMove & won == 0:
    computer = player + 1
    if computer > 3:
        computer = 1
```

`lastMove` 與 `won` 這兩個新的整數變數是代表電腦上一次的選擇以及是否獲勝。由於我只是想示範遊戲的一般性設計，所以我沒有把上述這段加入原本的程式中。

下一版的剪刀石頭布遊戲將不再使用鍵盤輸入與螢幕輸出，而是改用按鈕與 LED。

剪刀石頭布遊戲加裝開關與 LED

如果把遊戲移植到 Raspberry Pi 上應該會有趣又好玩，你可用按鈕來選擇要出什麼，並用 LED 來代表這回合是輸、贏或平手。在本專案中，我還會提到如何在 Raspberry Pi 上透過 Python 來處理中斷，在程式中加入了這樣一個警告機制。請在命令列輸入以下指令來啟動程式：

```
python prs_with_LEDs_and_Switches.py
```

請如圖 4-3 來設定 Raspberry Pi。

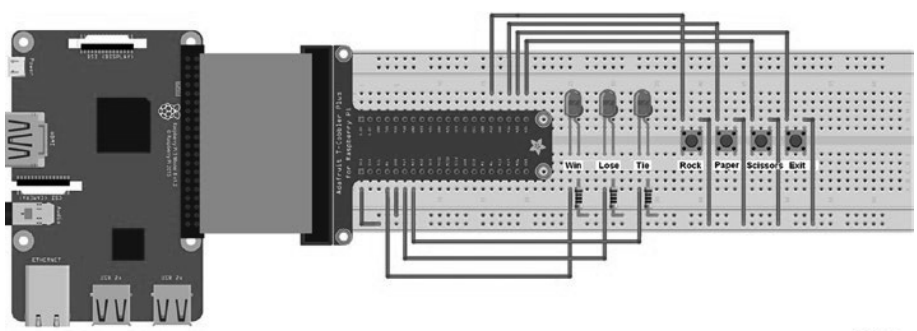

圖 4-3 剪刀石頭布遊戲機的 *Fritzing* 元件圖

自動化 Nim 遊戲的 Fritzing 硬體示意圖如圖 4-10。

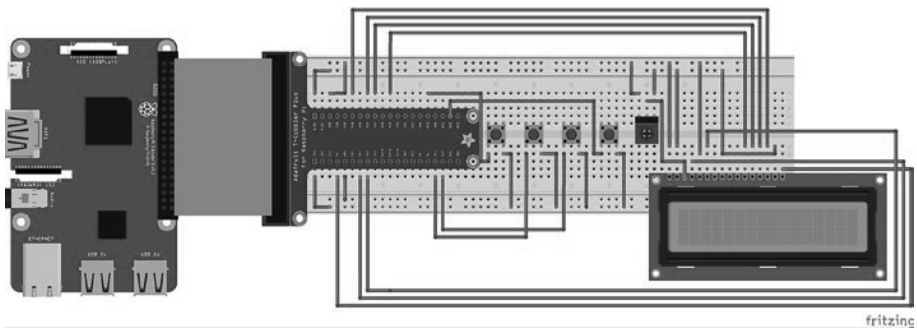

圖 4-10 自動化 *Nim* 遊戲的 *Fritzing* 硬體示意圖

這次的線路顯然複雜許多，不太容易要用一張 Fritzing 元件圖來表示。因此我除了提供 LCD 對 Pi Cobbler 擴充板接線的示意圖，以及包含了本系統所有腳位的接線清單。圖 4-11 是 LCD 模組與 Pi Cobbler 擴充板的接線示意圖。

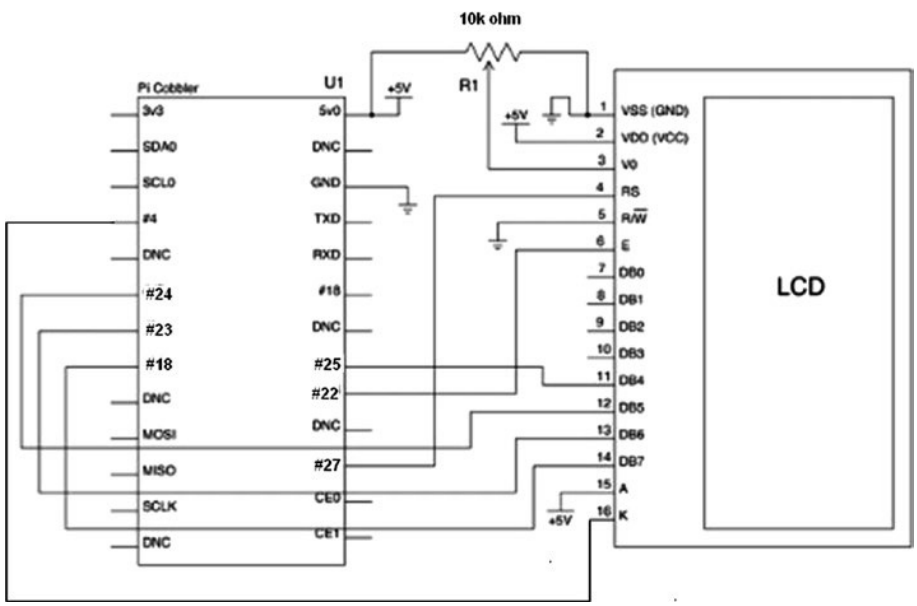

圖 4-11 *Pi Cobbler* 擴充板與 *LCD* 模組之接線示意圖

CHAPTER 5

模糊邏輯系統

本章將繼續探討第二章提到的模糊邏輯（FL）概念。在此會示範兩個模糊邏輯專題。第一個範例是我們偶爾會碰到的情況：上餐廳吃飯時如何計算小費。第二個範例則稍微複雜一點，要實作一個以 FL 為控制技術的冷熱控制系統。兩個範例都使用 Python 搭配 pyFuzzy 函式庫，後者是將 FL 整合進 Python 程式語言中。另外我還會介紹幾個新的 FL 主題。我把這些新主題搭配 FL 小費範例來說明，希望為這些新概念提供更好的框架。

在進入基礎模糊邏輯系統（fuzzy logic system, FLS）之前，你得先把 Raspberry Pi 設定好才能載入並執行 FL 相關的範例 5-1。

零件清單

本章範例所需零件如表 5-1。

表 5-1 零件清單

說明	數量	Remark
Pi Cobbler 腳位擴充板	1	40 腳位的版本，T 型或 DIP 型都可以
免焊麵包板	1	860 孔，並有電源供應軌
跳線	1 包	很多地方都能買到
LED	3	常用品，很多地方都能買到
220Ω 電阻	3	1/4 瓦特

範例 5-3：FLS 冷熱控制系統

本專題先假設你已經看過也理解了前面的第一個專題，關於 FL 的概念應該不需要再深入討論了。本專題在開發時也是遵循 FLS 演算法。並且本專題不會用到任何視覺化相關的程式碼，這在上個專題都運用過了。反之，讀者隨時可以把繪圖程式碼補回去來觀察系統的運作過程。

談到加熱、通風與冷卻系統（HVAC）時，一般來說是指一個扮演空調機或升溫器的加熱幫浦。圖 5-16 是這類系統的功能方塊圖，這樣的設定在控制學術語中也稱為閉迴路（*closed-loop*）系統。

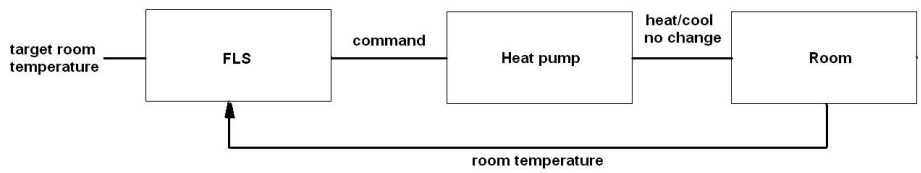

圖 5-16 *HVAC* 閉迴路系統

在此定義溫度 (t) 為明確輸入變數來代表要升溫或冷卻的室內溫度。一般而言，人們使用熱（*hot*）與冷（*cold*）這樣的詞彙作為對室內溫度的修辭。這些詞彙與其相關者可以發展成一組語意詞彙，例如：

```
T(t) = { cold, comfortable,  hot }
```

關於 *T(t)* 這個式子代表對於輸入變數 t 的拆解結果。本語意集拆解後的的每一個成員都是代表或關聯於一個數值性的溫度範圍。例如，*cold* 可以是 40°F 到 60°F，*hot* 則可能是 70°F 到 90°F。如果我們認為 20°F 是一個合適的區間值，也可以在範圍中間填入其他的語意詞彙。

有另一個名為目標溫度（*target temperature*）的輸入，這是由身處室內的人所設定的。這就好比是設定室內恆溫器。

圖 5-17 是這些隸屬函數，用於將明確、非模糊的室內與目標溫度值去對應到各自的模糊語意詞彙。在此只有顯示一組隸屬函數，但可被室內溫度與目標溫度這兩個輸入變數所共用。

CHAPTER 6

機器學習

本章從機器學習所囊括的各種主題開始介紹，我在第二章有稍微提過。機器學習現在不管是在產業界或學術界都是超熱門的主題。像是 Google、Amazon 與 Facebook 這樣的大公司已在機器學習上投資了數百萬美元來改良其產品與服務。我會從幾個相當簡單的 Raspberry Pi 範例來說明電腦在相當原始或早期的意義上如何「學習」。

首先，我想對從 Bert van Dam 的著作：*Artificial Intelligence: 23 Projects to Bring Your Microcontroller to Life*（Elektor Electronics Publishing，2009）中獲得關於本章的靈感與知識表達感謝之意。雖然 van Dam 沒有把 Raspberry Pi 當作微控制器來使用，但他所採用的概念與技術依然非常實用，在此特別感謝。

零件清單

第一個範例需要以下零件，列如表 6-1。

表 6-1 零件清單

說明	數量	說明
Pi Cobbler 擴充板	1	40 腳位版本，T 型或 DIP 型都可以
免焊麵包板	1	300 孔，並有電源供應軌
免焊麵包板	1	300 孔，不需電源供應軌

續下頁

A、B 的適應值範圍縮小之後，會讓 #2 抽取的適應值選擇結果由 C 變成了 D。這情況與先前的顏色選擇範例完全相同，每次按下按鈕都會改變決策點，因而改變了選擇兩個顏色的適應值範圍。

修改適應值範圍與後續如何選擇的策略可說是輪盤演算法的重要基礎。你很快就會學到，要在小型移動式機器人這類自動車上做到學習行為的話，這套演算法相當管用。例如，輪盤演算法已被應用在染色體存活統計上的用藥研究上。

範例 6-2：全自動機器人

來看看 Alfie，這是我為下圖這台小型自動移動式機器人所取的名字。圖 6-7 是 Alfie 的照片：

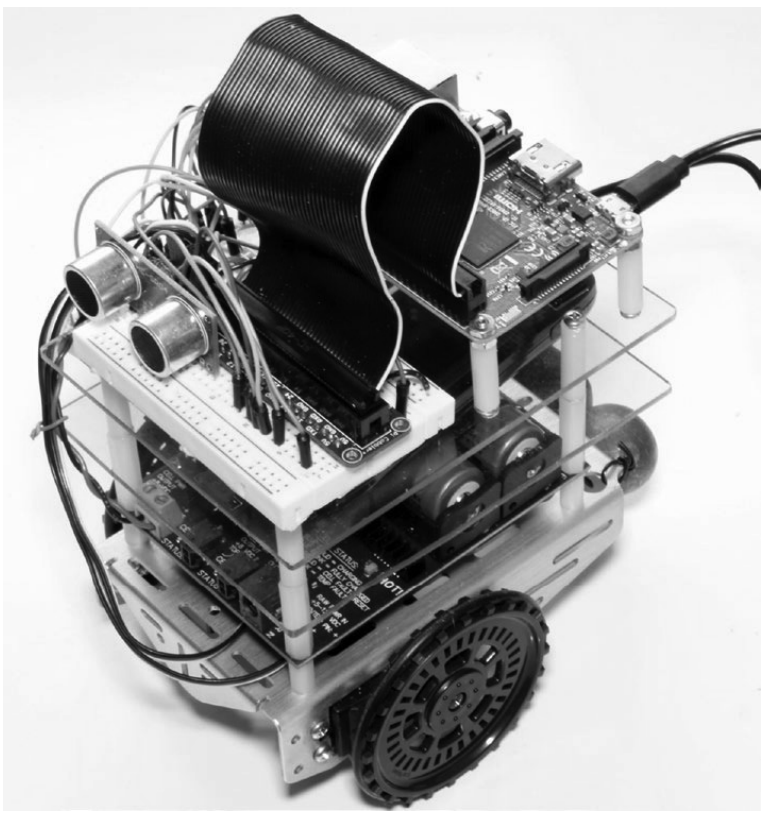

圖 6-7 *Alfie* 機器人

CHAPTER 7

類神經網路

本章要繼續探討機器學習並集中火力在類神經網路（Artificial Neural Network, ANN）。本章有幾個範例靈感是來自於 Bert van Dam，再次感謝他。

零件清單

範例 7-1 會用上一章的 Alfie 機器小車與額外一些零件，整理如表 7-1。

表 7-1 零件清單

說明	數量	說明
Pi Cobbler 擴充板	1	40 腳位版本，T 型或 DIP 型都可以
免焊麵包板	1	700 孔，並有電源供應軌
跳線	1 包	
超音波感測器	1	HC-SR04
4.9kΩ 電阻	2	1/4 瓦特
10kΩ 電阻	6	1/4 瓦特
MCP3008	1	8 通道 ADC 晶片，DIP 型
光電管	1	硫化鎘（CdS）型都可以

讓我們從所有 ANN 中最簡單的 Hopfield 網路開始介紹。

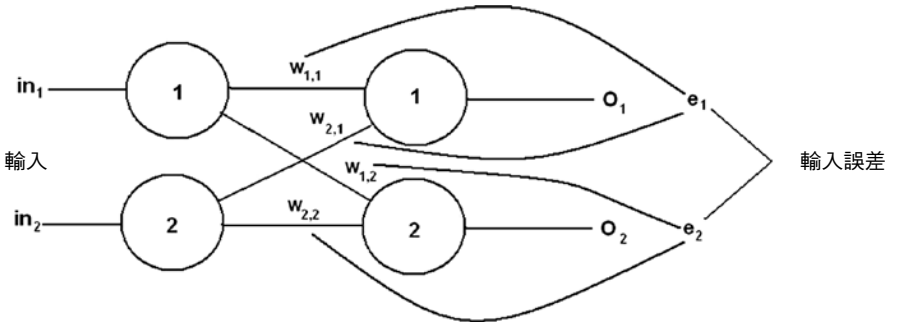

圖 8-10 多個輸出節點的誤差分配問題

結果證明，多節點與單一節點的做法完全相同。這點倒是沒錯，因為輸出節點都是獨立的，彼此間沒有互聯。如果不是這樣的話，就很難做到在彼此連結的輸出節點之間做到後向傳播。

分配誤差方程式相當簡單，根據連結於輸出節點的權重比例就能算出。例如要判斷圖 8-10 中的 e_1 修正值，應用於 $w_{1,1}$ 與 $w_{2,1}$ 的分數部分如下：

$$w_{1,1}/(w_{1,1} + w_{2,1}) \text{ 與 } w_{2,1}/(w_{1,1} + w_{2,1})$$

同樣地，以下是 e2 的誤差。

$$w_{1,2}/(w_{1,2} + w_{2,2}) \text{ 與 } w_{2,2}/(w_{1,2} + w_{2,2})$$

到目前為止根據輸出誤差來調整權重的過程還算簡單。因為訓練資料中有正確解答所以要很容易判斷誤差。對於兩層的 ANN 來說，這樣就足夠了。但如果是三層 ANN，其中的隱藏層輸出可說一定會產生誤差，並且沒有可用的訓練資料來判斷誤差值，這時應該怎麼辦呢？

三層 ANN 的後向傳播

圖 8-11 是一個三層六節點的 ANN，每層各有兩個節點。在此我刻意簡化了這個 ANN，這樣要理解網路所需的有限後向傳播就更容易了。

以下使用 Python 來實作這個誤差計算函式：

```
# y = mx + b
# m 為斜率 , b 為 y 軸截距
def computeErrorForLineGivenPoints(b, m, points):
    totalError = 0
    for i in range(0, len(points)):
        totalError += (points[i].y - (m * points[i].x + b)) ** 2
    return totalError / float(len(points))
```

以下是程式中所用的正式誤差計算方程式：

$$e_{m,b} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(m * x_i + b\right)\right)^2$$

產生最佳配適度的斜線（由上述的誤差函數求得）就是位於整體資料集所可能產生的最小值處。在此的技巧是建立特定格式的誤差函式來算出合適的 m 與 b 值以達到整體最小值。在開始之前，先把 m、b 與 $e_{m,b}$ 彼此的關係以視覺化方式呈現應會更容易理解。圖 8-14 引用自 Matt 部落格，可以清楚看出變數之間的捲曲關係。

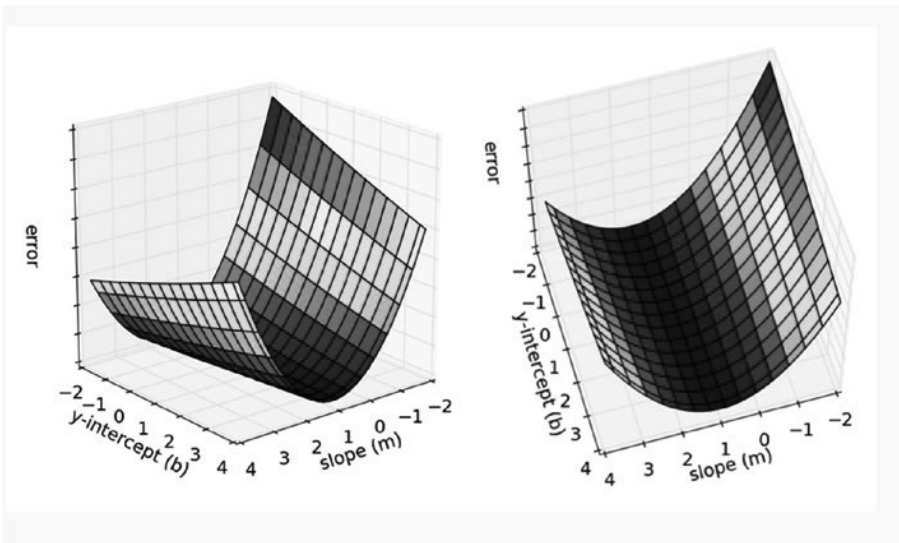

圖 8-14 b 與 $e_{m,b}$ 的圖表

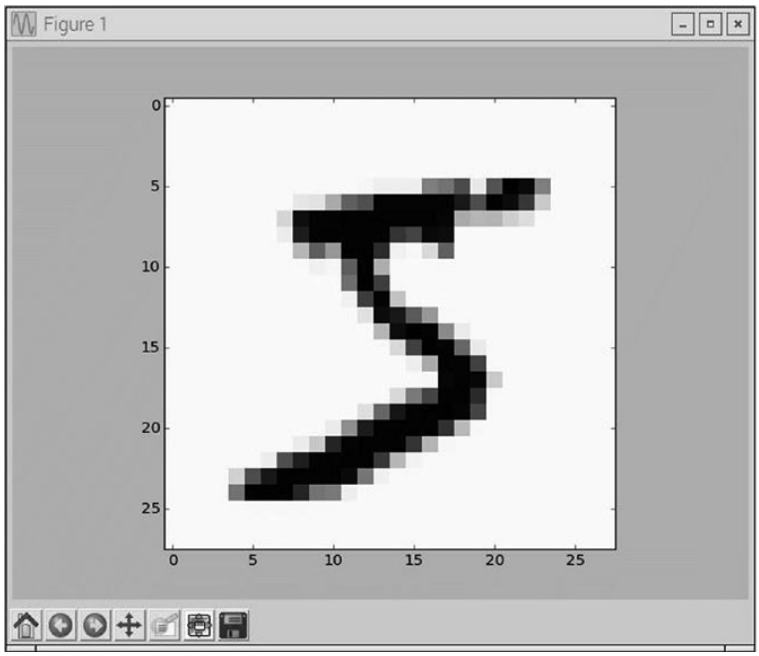

圖 9-5 數字的圖像

你可以發現，這個圖像顯示了隨意手寫的羅馬數字 5，這是在眾多紀錄資料中其中一個被拿來訓練 ANN 的紀錄。附帶一提，若你回去看圖 9-1，你可以看到數字 5 正是第一組紀錄的符號，這也就確核了資料身份。

接下來，我要將議題轉向如何準備資料集，以讓它們有效地被 ANN 使用。

調整輸入與輸出的資料集

你已經知道 MNIST 資料集中的所有數值範圍都必須介於 0 到 255 之間，這大大超過了我所開發的任何 ANN 所能接受的範圍。輸入值的理想範圍是從 0.01 到 1.0，恰好符合 S 型函數的輸入值條件。以下的 Python 程式會將 MNIST 資料集的數值換算，調整成 ANN 所能接受的數值範圍。

```
adjustedRecord0 = (np.asfarray(record0[1:]) / 255.0 * 0.99) + 0.01
```