前言

機器學習簡史

機器學習是人工智慧（AI）的子領域，在這領域裡，電腦從數據中學習目標行為，而不是靠明確的程式設計來達成——這些目標通常是為了改善他們在一些特定任務上的表現成效。**機器學習**這個術語是 1959 年初創造出來的（源自 Arthur Samuel，人工智慧領域的傳奇），但在二十一世紀，機器學習幾乎沒有什麼重要的商業成功案例。該領域仍然屬於學術界利基型的研究領域。

在 20 世紀 60 年代，AI 社區中許多人對於人工智慧的未來過於樂觀。當時的研究人員（如 Herbert Simon 和 Marvin Minsky）聲稱人工智慧將在幾十年內達到人類智能的水平[1]。

> 機器將在二十年內，有能力完成人類所有會做的工作。
>
> — Herbert Simon, 1965

> 三到八年內，我們將擁有一台具有與人類同等智慧的機器。
>
> — Marvin Minsky, 1970

被樂觀態度沖昏頭的研究人員專注於所謂的**強人工智慧**（*strong AI*）或**通用人工智慧**（*general artificial intelligence, AGI*）項目，試圖建立 AI 代理人，而這些代理人具有解決問題、知識表達、學習和計劃，自然語言處理、感知和運動控制的能力。這種樂觀氛

1 這樣的觀點是受到 Stanley Kubrick 於 1968 年在 *2001：A Space Odyssey* 中構建了 AI 代理人 HAL 9000 的啟發。

碁峰 www.gotop.com.tw

圍有助於吸引大量資金從國防部等主要投資者，進入新的研究領域，但是這些研究人員試圖解決問題的野心過大，而最終注定失敗收場。

從學術界應用到產業界，人工智慧的研究鮮少有任何跳躍式的進展，加上一系列所謂的人工智慧寒冬緊隨其後。在這些人工智慧寒冬（基於冷戰期間核冬天作為類比）下，對人工智慧感興趣的人與投注的資金更加減少。偶爾伴隨著人工智慧技術成熟曲線發生炒作，但持續力道都很短。到 20 世紀 90 年代初，對人工智慧感興趣的人與投注的資金更是跌到了谷底。

人工智慧回來了！但為什麼是現在？

在過去的二十年裡，人工智慧再度掘起，從純粹的學術領域為開端，至今成為蓬勃發展的領域，吸引無數學術及企業中的頂尖人才投入其中。

有三個關鍵的發展導致了人工智慧的再次興起：機器學習演算法的突破，可取得的大量數據和超高速電腦。

首先，研究人員不再專注過於野心勃勃的強人工智慧專案，而將注意力轉到較為明確而特定的強人工智慧子問題，也稱為**弱人工智慧**（*weak AI*）或**狹義人工智慧**（*narrow AI*）。針對改善特定範圍任務的解決方案，進一步帶來了演算法上的突破，因而為成功的商業應用做好了準備。許多演算法（通常最初是在大學或私人的研究實驗室開發出來）迅速地被開源出來，進而加快這些技術被產業所採用。

其次，數據擷取成為大多數組織關注的焦點，以及歸功於數據儲存技術的進步，儲存成本急劇下降。得力於網際網路，大量數據以前所未有的規模被廣泛且公開地取得。

第三，電腦變得越來越強大並且可以在雲端上被使用，讓 AI 研究人員可以輕易且低成本地擴展 IT 基礎架構，而不需要對硬體進行大量的前期投資。

AI 應用的出現

這三種因素將人工智慧從學術界推向了產業界，吸引每年越來越多的人關注以及更多的資金。AI 不再僅僅是理論上的興趣領域，而是一個蓬勃發展的應用領域。圖 P-1 呈現由 Google Trends 查詢到的一張圖表，該圖表指出過去五年對機器學習領域感興趣的成長趨勢。

碁峰 www.gotop.com.tw

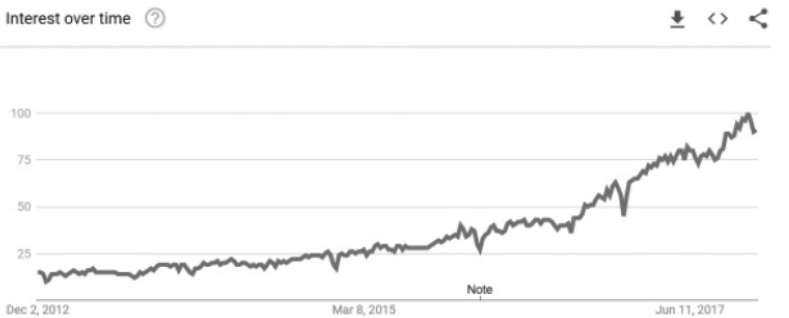

圖 P-1　隨時間對機器學習感興趣趨勢

如今 AI 被視為一種突破性且在各領域被廣泛應用的技術，就像電腦和智慧手機的出現一般。它將在未來十年對每一個產業產生重大影響 [2]。

涉及機器學習的成功商業應用包括（但不僅限於此）光學字元識別、垃圾郵件過濾、圖像分類、電腦視覺、語音識別、機器翻譯、分組與分群、合成數據的生成、異常偵測、網絡犯罪預防、信用卡詐欺偵測、網絡詐欺偵測、時間序列預測、自然語言處理、棋盤遊戲和電玩遊戲、文件分類、推薦系統、搜索、機器人、線上廣告、情感分析，DNA 測序，金融市場分析，訊息檢索、問答和醫療照護決策制定。

AI 應用在過去 20 年的重要里程碑

以下列出的里程碑，推動了人工智慧從一個主要為學術討論的主題，成為當今主流的技術。

- 1997 年：深藍（Deep Blue），一個自 20 世紀 80 年代中期以來一直在開發的 AI 機器人，在一場高度宣傳的國際象棋比賽中擊敗了國際象棋冠軍 Garry Kasparov。
- 2004 年：DARPA 推出 DARPA 大挑戰賽，賽事年年舉辦，主要是挑戰在沙漠中進行自動駕駛。2005 年，史丹佛大學獲得了最高獎項。2007 年，卡內基梅隆大學在城市環境中締造了自動駕駛的壯舉。2009 年，Google 製造了一款自動駕駛汽車。到了 2015 年，許多主要技術巨頭（包括 Tesla、Waymo 和 Uber）已經開始以建造主流自動駕駛技術為目標，並且受到完善投資的專案項目。

2　根據麥肯錫全球研究所的研究，到了 2055 年，人類有超過一半的專業活動可以做到自動化。

碁峰 www.gotop.com.tw

- 2006 年：多倫多大學的 Geoffrey Hinton 引入了快速學習演算法，用來訓練多層神經網路，開啟了深度學習的革命。
- 2006 年：Netflix 發起了以 100 萬美元為獎金的 Netflix Prize 競賽，考驗參賽團隊使用機器學習來改善其推薦系統的準確度至少 10%。最終，其中一個團隊在 2009 年獲得該獎項。
- 2007 年：由 Alberta University 的一支團隊所實作的人工智慧，在西洋跳棋上展現了超越人類的表現。
- 2010 年：ImageNet 推出年度競賽——ImageNet Large Scale Visual Recognition Challenge（ILSVRC），針對大規模且已經被完善處理的圖片集，參與團隊需要使用機器學習演算法，來進行正確地偵測和分類。這引起了學術界和科技巨頭的極大關注。得力於深層卷積神經網絡的進步，分類錯誤率從 2011 年的 25% 下降到 2015 年的幾個百分點。這促成了電腦視覺和物體識別的商業應用。
- 2010 年：微軟推出適用於 Xbox 360 的 Kinect。由微軟研究院的電腦視覺團隊所開發的 Kinect，它能夠追蹤人體運動並將其轉化為遊戲。
- 2010 年：Siri 作為首批主流數位語音助理之一，被 Apple 收購，並且在 2011 年 10 月作為 iPhone 4S 的一部分一起發布。最終，Siri 被發布到 Apple 的所有產品。受卷積神經網路和長短期記憶遞歸神經網路的特性加持，Siri 得以處理語音識別和自然語言處理兩種任務。最終，Amazon、Microsoft 和 Google 都加入了戰局，相應地釋出 Alexa（2014）、Cortana（2014）和 Google Assistant（2016）。
- 2011 年：IBM Watson，是由 David Ferrucci 所帶領的團隊開發出來的問答型人工智慧代理人，且該代理人在 *Jeopardy!* 擊敗了前任的優勝者 Brad Rutter 和 Ken Jennings。IBM Watson 現在應用於數種行業，包括醫療照護和零售。
- 2012 年：由 Andrew Ng 和 Jeff Dean 領導的 Google Brain 團隊所訓練的神經網路能夠透過觀看 YouTube 影片，從其中無標記的圖像來識別貓。
- 2013 年：Google 贏得了 DARPA's Robotics Challenge，該挑戰中半自動駕駛機器人在危險的環境下，進行了多樣複雜的任務，例如駕駛車輛、穿過瓦礫、從堵塞的入口處移走碎石、打開一扇門和爬梯子。
- 2014 年：Facebook 公開了 DeepFace，這是一個基於神經網路的臉部識別系統並且具有 97% 的精確度。這樣的精確度逼近人類的表現，而且比以前的系統效能提高了 27% 以上。

碁峰 www.gotop.com.tw

- 2015 年：人工智慧成為主流，並且普遍地出現在媒體上。
- 2015 年：Google DeepMind 的 AlphaGo，在圍棋比賽中擊敗了世界級的專家樊麾。2016 年，AlphaGo 擊敗李世石，2017 年，AlphaGo 擊敗了柯潔。2017 年，AlphaGo Zero 對前版本的 AlphaGo 連勝 100 場。AlphaGo Zero 採用無監督學習技巧透過與自己對奕而成為圍棋大師。
- 2016 年：Google 針對 Google Translate 進行重大改進，用深度學習的機器翻譯系統替換了現有的片語式翻譯系統，不僅減少高達 87% 的翻譯錯誤率，也接近人類的精確度。
- 2017 年：由 Carnegie Mellon 開發的 Libratus 在無限注一對一的德州撲克中獲勝。
- 2017 年：OpenAI 訓練的機器人在 Dota 2 錦標賽中擊敗了職業選手。

從狹義人工智慧到通用人工智慧（AGI）

當然，將 AI 成功應用於特定且明確的問題只是一個開始。人工智慧社區越來越相信可以透過結合幾個弱 AI 系統，繼而開發出強人工智慧。這個強人工智慧或 AGI 代理人將能夠在許多廣泛目標任務中達到人類水平的表現。

人工智慧達到人類水平表現後不久，一些研究人員預測強人工智慧將超越人類的智慧並達到所謂的**超級智能**（*superintelligence*）。欲實現這種超級智能，預估至少要 15 年到 100 年後，但大多數研究人員認為人工智慧在未來的數個世代裡將獲得充份的進展，以達到超級智能的水平。這是否只是再次的膨脹炒作（就像我們之前所見的 AI 循環），或者這次真是不同於之前？

就讓我們拭目以待。

目標和方法

迄今為止，大多數成功的商業應用領域（如電腦視覺、語音識別、機器翻譯和自然語言處理）都是利用有標籤的資料集，進行監督式學習。然而，世界上大部分數據都**沒有標籤**。

碁峰 www.gotop.com.tw

在本書中，我們將包含**非監督式學習**（這是機器學習領域的分支，用來查找資料內隱藏的樣式）並學習無標籤資料內潛藏的結構。根據許多產業專家，如 Yann LeCun，Facebook 的 AI 研究院的處長和紐約大學的教授，認為非監督式學習是人工智慧的下一個前沿技術，並且可能是 AGI 的關鍵。基於這個原因和其他許多理由，非監督式學習是當今人工智慧中最流行的主題之一。

本書的目標是概述所需的概念和工具，讓你能夠將這項技術應用於需要透過人工智慧處理的日常問題上。換句話說，這是一本教會你如何應用的書籍，讓你建構可以落地的系統。我們還將探索如何有效地標記無標記的資料集，以便轉換非監督式學習問題成為半監督式學習的問題。

本書將採用實際操作範例的方法介紹一些理論，但重點關注在應用非監督式學習技術來解決實務上的問題。你可以在 GitHub 上取得以 Jupyter Notebook 形式的資料集和程式碼（*http://bit.ly/2Gd4v7e*）。

一旦你具備了概念性理解和實踐經驗，你將能夠應用非監督式學習於大量且無標籤的資料集，以便發現隱藏的樣式、獲得更深入的業務洞察力、偵測異常，基於相似性分類群組、執行自動特徵工程和特徵選擇、生成合成資料集等更多的應用上。

前提

本書假設你已具有一些 Python 的程式經驗，包括熟悉 NumPy 和 Pandas。

有關 Python 的更多訊息，可造訪 Python 官方網站（*https://www.python.org/*）。若想更了解 Jupyter Notebook，請造訪 Jupyter 官方網站（*http://jupyter.org/index.html*）。想要複習關於大學程度的微積分、線性代數、機率和統計，可閱讀由 Ian Goodfellow 和 Yoshua Bengio 所撰寫的《*Deep Learning*》textbook 第一部分（*http://www.deeplearningbook.org/*），繁體中文版《深度學習》由碁峰資訊出版。想要複習關於機器學習，請閱讀《*The Elements of Statistical Learning*》（*https://stanford.io/2Tju4al*）。

本書架構

本書分成四個部分，涵蓋以下主題：

碁峰 www.gotop.com.tw

第一部分，非監督式學習的基礎

監督式學習與非監督式學習的差異之處、受歡迎的監督式與非監督式演算法概觀以及完整的機器學習專案。

第二部分，使用 *Scikit-Learn* 開發非監督式學習

維度縮減、異常偵測和分群與群組區隔。

更多關於第一部分和第二部分討論的概念，請參閱 Scikit-learn 文件（*https://scikit-learn.org/stable/modules/classes.html*）。

第三部分，使用 *TensorFlow* 和 *Keras* 開發非監督式學習模型

特徵學習和自動特徵萃取、自動編碼器和半監督式學習。

第四部分，使用 *TensorFlow* 和 *Keras* 開發非監督式深度學習模型

受限波爾茲曼機、深度信念網路和對抗生成網路。

本書編排慣例

以下為本書使用的編排規則：

斜體字（*Italic*）

表示新名詞、超連結、電子郵件位址、檔名以及副檔名。中文以楷體表示。

定寬字（`Constant width`）

用於程式原始碼，以及篇幅中參照到的程式元素，如變數、函式名稱、資料庫、資料型態、環境變數、程式碼語句以及關鍵字等。

定寬粗體字（**`Constant width bold`**）

代表使用者輸入的指令或文字。

定寬斜體字（*`Constant width italic`*）

代表需要配合使用者提供的變數，或者是使用者環境來更換的文字。

碁峰 www.gotop.com.tw

第一章

機器學習領域裡的非監督式學習

大多數的人類和動物的學習過程都是非監督式學習。以蛋糕來比喻，非監督式學習就像那塊蛋糕，監督式學習就像撒於蛋糕上的糖霜，而強化學習則是蛋糕上的櫻桃。我們知道如何製造糖霜和櫻桃，但卻不知道如何製造蛋糕。在開始深入真正的人工智慧之前，我們需要解決非監督式學習的問題。

— Yann LeCun

在本章中，我們將探索規則式系統和機器學習之間的差異、監督式學習和非監督式學習之間的差異，以及它們之間相對的優點與缺點。

我們還將討論到許多受歡迎的監督式學習演算法和非監督式學習演算法，並且簡潔地檢視強化學習與半監督式學習，如何融入其中。

機器學習的基礎術語

在一頭栽入不同型態的機器學習之前，我們先來了解一個簡單且常見的例子 : 垃圾郵件過濾，以便讓即將介紹的概念更加具體。我們需要撰寫一個程式，該程式可以收信，並且將信件正確地分類為「垃圾郵件（spam）」和「非垃圾郵件（not spam）」。這是一個十分明確的分類問題。

碁峰 www.gotop.com.tw

作為複習，這裡列出一些機器學習的術語：這個範例的**輸入變數**（*input variable*）是信件內的文字內容，這些輸入變數也就是所謂的**特徵**（*features*）、**預報器**（*predictors*），或者是**獨立變數**（*independent variables*）。而**輸出變數**（*output variable*），即是我們試著進行預測的結果，就是「spam」或是「not spam」兩種**標籤**。它也稱之為**目標變數**（*target variable*）、**應變數**（*dependent variable*），或者是**響應變數**（*response variable*）（也稱為**類別**），因為這是一個分類問題。

用於訓練 AI 的範例資料集稱為**訓練集**，而每一筆範例資料都稱為一個訓練**實例**（*instance*）或**樣本**（*sample*）。在訓練過程中，AI 持續嘗試最小化**成本函數**（*cost function*）或**錯誤率**（*error rate*），更積極地解釋是，最大化它的**價值函數**（*value function*），以這個例子來說，就是正確分類電子郵件的比例。人工智慧積極地在訓練過程中最小化錯誤率，來進行優化，而錯誤率是經由比較人工智慧進行預測的標籤和實際的標籤計算得出。

然而，我們最關心的是訓練好的 AI 模型對從未見過的電子郵件泛化程度會有多好。這將會是 AI 模型的真正考驗：藉由訓練集樣本訓練所學得的模型，它能否正確分類以前從未見過的電子郵件？這個**泛化誤差**（*generalization error*）或**樣本外誤差**（*out-of-sample error*）是我們主要用來評估機器學習解決方案的方式。

這組未被模型見過的樣本稱為**測試集**（*test set*）或**留出集**（*holdout set*）（因為資料從訓練集中被保留出來）。如果我們選擇有多個留出集（或許可以用來衡量所訓練模型的泛化誤差，這是個可行的方法），我們可以保留一些訓練過程中使用的留出集，在最終測試集進行驗證前，對訓練的進度進行評估；這些訓練過程中用來評估進度的留出集稱為**驗證集**（*validation sets*）。

我們將上述用於訓練 AI 模型所提及的概念整理一下，AI 模型基於訓練資料（**經驗**）進行訓練，以改善標記垃圾郵件（**任務**）的錯誤率（**效能**），而最終的成功標準是模型基於學得的經驗，在新的且從未見過的資料集下的泛用程度（**泛化誤差**）。

基於規則式 vs. 機器學習

使用基於規則的方法，我們可以設計一個帶有明確規則的垃圾郵件過濾器來濾出垃圾郵件，比方說帶有旗幟標記的電子郵件中有「u」而不是「you」、「4」而不是「for」、「BUY NOW」等。然而，假如有心人士改變他們的垃圾郵件行為以規避這些郵件過濾的規則，那麼系統將很難被維護。如果我們使用基於規則的系統，我們必須經常手動調

碁峰 www.gotop.com.tw

整規則以保持最新的狀態，而且窮舉所有需要的規則來確保系統運作良好，將是十分昂貴的。

如果不使用基於規則的方法，取而代之的是我們可以基於電子郵件資料，並且使用機器學習來訓練模型，讓該模型能夠自動地習得郵件規則並且正確地標記出垃圾郵件。這個基於機器學習的系統也可以隨著時間的推移自動調整規則，這種系統的訓練和維護成本將會便宜得多。

在這個簡單的電子郵件問題中，我們是有可能用人工的方式製作規則的，但是對於很多問題來說，人工製作規則根本不可行。例如，設計自駕車，你可以想像一下如何為汽車遇到的每一個狀況製作規則。這是一個棘手的問題，除非汽車可以基於自己的經驗，進行學習並且調適規則。

我們還可以使用機器學習系統作為探索或數據探勘的工具，以便更深入了解我們正在努力解決的問題。例如，在電子郵件垃圾郵件過濾器範例中，我們可以了解哪些單字或片語是垃圾郵件中最具預測性的特徵，以及識別新出現的惡意垃圾郵件樣式。

監督式學習 vs. 非監督式學習

機器學習領域有**監督式學習**和**非監督式學習**兩個主要分支，以及許多用來連接這兩種主要分支的子分支。

在監督式學習中，AI 代理人藉由標籤來改進在某些任務上的表現。在垃圾電子郵件過濾器問題中，我們有一個含有所有電子郵件內容的資料集也知道其中的哪些電子郵件是否為垃圾郵件（這就是所謂的**標籤**（*labels*）），這些標籤在幫助監督式學習 AI 將垃圾郵件與其他電子郵件分開這方面上非常有價值。

在非監督式學習中，標籤是無從取得的。因此，AI 代理人的任務是沒有明確定義的，並且表現也無法如此清晰地被衡量。試想一下垃圾電子郵件的過濾問題——而這次並沒有標籤。現在 AI 代理人將試圖了解電子郵件的基礎結構，並且將電子郵件數據庫中的郵件分成不同的群組，使得群組內的電子郵件彼此相似，但是又與其他群組中的電子郵件截然不同。

這種非監督式學習的問題定義不如監督式學習來得清楚，而且更難被 AI 代理人所解決。但是如果能夠處理得當，該解決方案將更具威力。

碁峰 www.gotop.com.tw

原因如下：非監督式學習 AI 可能會找到後來被標記為垃圾郵件的數種群組——但 AI 也可能會找到後來被標記為「重要」的群組，抑是可以歸類為「家庭」、「專業」、「新聞」、「購物」等群組。換句話說，因為問題沒有嚴格定義任務，因此 AI 代理人可能會找到超出我們最初感興趣的樣式。

此外，在發掘未來資料中的新模式時，非監督式系統比監督式系統更好，這樣的特性使非監督式解決方案在未知的基礎上更加靈活。這就是非監督式學習的威力。

監督式學習的優缺點

監督式學習擅長於在具有明確定義與完整標籤的任務中，進行模型效能的優化。例如，有一個非常大量且具有標籤的圖像數據集。如果數據集的圖像數量的確夠多，而且也有足夠算力的電腦，讓我們可以使用適合的機器學習演算法（即卷積神經網絡，CNN），便可以建立一個基於監督式學習且具有優良效能的圖像分類系統。

由於進行監督式學習模型建立時，監督式學習能透過比較預測的圖像標籤與實際的圖像標籤計算成本函數，來最小化成本函數，因此，它可以在未曾見過的圖像（來自留出集）上，盡可能地降低預測誤差。

標籤如此有用的理由，是因為它能夠在模型訓練過程中作為誤差評量的依據，進而提升模型的訓練結果。如果沒有這樣的標籤，就無從得知被訓練的模型是否成功（或不成功）地分類圖像。

然而，人工地對圖像數據集進行標記的成本很高，即便是在已經特別精煉過的數據集情況下，成本依舊不低。這是一個麻煩的問題，因為監督式學習非常擅長於對有標籤的圖像進行分類，但是對沒有標籤的圖像分類能力卻很差。

監督式學習在基於有標籤數據的強大表現，也同時地曝露出它在未曾見過的數據集上，模型泛化能力的侷限性。因為世界上大多數的數據是沒有標籤的，所以基於監督式學習的人工智慧，對於未知數據集的表現能力將受到相當的限制。

換句話說，監督式學習對於解決狹義人工智慧（narrow AI）問題非常有用，但並不適合用於解決較為進階且定義相對不明確的強人工智慧（strong AI）問題。

碁峰 www.gotop.com.tw

非監督式學習的優缺點

監督式學習在數據可取得、充足、具完善標籤以及所具備樣式穩定且明確的窄域任務下，訓練的效果凌駕於非監督式學習之上。

然而，對於數據的樣式不停地變化或不清楚，抑或沒有充足且具標籤的數據集情況下，非監督式學習則具有明顯的優勢。

非監督式學習不是透過標籤的指引，而是從它所使用的訓練數據集中學習隱含的資料樣式。它藉由一組數量明顯少於訓練數據集的參數組合，來試著表示訓練數據集的資料。透過執行這種表徵學習，非監督式學習能夠識別出數據中不同的樣式。

以影像數據集（此時沒有標籤）為例，非監督式學習可以基於影像彼此之間的相似程度與不同程度來識別和分組影像。例如，所有看起來像椅子的影像將被組合在一起，而所有看起來像狗的影像將被分組在一起等。

當然，非監督式學習並不能將這些組合，各自標記為「椅子」或「狗」，但現在相似的影像被組合在一起，為資料進行標籤的任務變得更為簡單了。我們可以直接對不同分組的圖像直接整批打上標籤，而不再是對數以百萬計的圖像逐一貼上標籤。

歷經一開始的訓練後，如果非監督式學習發現有影像不屬於任何組別，它便將這些無法被成功分組的影像創建單獨的組別，讓人類替這些尚未有標籤的新群組，給定適合的新標籤。

非監督式學習使以前棘手的問題更容易被解決，而且在可取得的歷史數據與未來的數據中，可以更加有效地找到隱藏的樣式。最重要的是，現在我們有一種人工智慧的方法，來協助我們從大量無標籤的資料中探索有價值的寶藏。

在解決明確而具體的問題上，即使非監督式學習不如監督式學習，但非監督式學習卻能更好地解決開放的強人工智慧（strong AI）問題和有效地泛化訓練得到的知識。

同樣重要的是，非監督式學習可以解決許多數據科學家在構建機器學習解決方案時遇到的常見問題。

碁峰 www.gotop.com.tw

使用非監督式學習改善機器學習解決方案

最近在機器學習方面的許多成功案例，得歸功於大量可取得的數據、電腦硬體和雲資源的進步，以及機器學習演算法的突破。但這些成功大多是狹義人工智慧（narrow AI）問題，比方說影像分類、電腦視覺、語音識別，自然語言處理和機器翻譯。

為了解決更有挑戰的的人工智慧問題，我們需要解放非監督式學習的價值。讓我們探索資料科學家構建解決方案時最常面臨的挑戰，以及非監督式學習如何提供幫助。

不充足的無標籤數據

> 我認為 AI 類似於建造一艘火箭。你需要一個巨大的引擎和大量的燃料。如果你有一個大型引擎和少量燃料，火箭將無法進入軌道。如果你有一個小發動機和大量燃料，火箭甚至無法升空。要製造火箭你需要一個巨大的發動機和大量的燃料。
>
> —吳恩達

如果機器學習是火箭飛船，資料將是燃料。沒有極大量的資料，火箭飛船是無法飛行的。但並非所有被製造出來的資料都是平等的。使用監督式學習演算法，我們需要大量有標籤資料，生成這些資料既困難又昂貴[1]。

透過非監督式學習，我們可以自動為無標籤的範例資料加上標籤。原理是我們將所有範例資料分群，然後使用同群中已被標記範例資料的標籤，為沒有標籤的範例資料賦予相同的標籤。無標籤的範例資料會使用與它們最相似範例資料的標籤作為標籤。我們將在第五章探索分群的議題。

過度擬合（Overfitting）

如果機器學習演算法從訓練資料中學會了過於複雜的函數，該機器學習模型可能會對留出集（如驗證集或測試集）中未見過的資料範例，作出很糟的預測結果。這是因為機器學習演算法擷取了太多資料中的噪音，進而過度擬合於訓練資料，因此在泛化的表現上，變得很差勁。換句話說，該演算法正在記住訓練資料而不是學習如何基於資料來概括知識[2]。

---

1 有一些新創公司（如 Figure Eight）提供人機混合（*human-in-the-loop*）的服務。

2 低度擬合（underfitting）是構建機器學習應用時，可能出現的另一個問題，但這問題更容易被解決。低度擬合是因為模型過於簡單，換句話說，採用的演算法無法建構出足夠複雜的函數，來貼近資料實際的分佈，以致於無法對目標任務做出好的預測和判斷。為了解決這個問題，我們可以增加這個演算法的複雜度（有更多參數，執行更多訓練迭代等），或使用更複雜機器學習演算法。

碁峰 www.gotop.com.tw

為了解決這個問題，我們可以使用非監督式學習作為**正規化器**（*regularizer*）。**正規化**（*regularization*）是一個用於降低機器學習演算法複雜性的流程，正規化協助機器學習演算法捕捉數據中的信號，但降低對於資料噪音的過度調適。採用非監督式學習進行預訓練是正規化的其中一種作法。採用非監督式學習重新產生原始資料的表示方法，然後再用監督式學習演算法進行訓練，而不像原先直接採用原始數據的方式進行訓練。

這種新的資料表示方式捕捉了原始資料的重要特性——資料真正隱含的特徵結構——同時忽略一些資料中不那麼具代表性的噪音。當我們提供這種新的資料表示方法給監督式學習演算法，它將可以面對較少的資料噪音並擷取更多有用的特徵，從而改善泛化錯誤。我們將在第 7 章探討特徵提取。

高維度災難（Curse of dimensionality）

即使電腦運算能力有所提高，利用機器學習演算法處理大數據仍然是個難題。一般來說，增加更多的運算單元並不會造成太大問題，因為我們可以使用 map-reduce 解決方案來進行平行化運算操作。但是，隨著具有的特徵越多，模型訓練就越顯困難。

在一個非常高維的空間中，監督式演算法需要學習如何分類空間中的點並構建近似函數，以便做出正確的決策。當特徵非常多的時候，不管從運算時間或運算資源角度上來看，這種搜尋都非常昂貴。在某些情況下，可能無法有效率地找到一個好的解決方案。

這個問題被稱為**高維度災難**（*curse of dimensionality*），非監督式學習很適合被用來解決和改善這個問題。透過降維，我們可以找到原始特徵集的顯著特徵，將維度減少到可處理的大小，並且在這過程中丟失很少的重要訊息，然後使用監督式演算法來更有效地找到近似函數。我們將在第 3 章介紹降維。

特徵工程

特徵工程是資料科學家處理資料時最重要的任務之一。如果沒有正確的特徵，機器學習演算法將無法有效地從特徵空間中分別出未曾見過的資料點，進而作出好的分類。然而，特徵工程通常是非常勞動密集型；它需要人介入其中，並且具創造性地標出正確的特徵類型。取而代之的是，我們可以基於非監督式演算法來進行表徵學習，來幫助我們完成特徵工程的任務。我們將在第七章探討自動特徵擷取。

碁峰 www.gotop.com.tw

離群值

資料品質也是非常重要。如果機器學習演算法基於罕見且扭曲的離群值進行訓練，產生出的模型的泛化誤差表現，將低於忽略或分別對異常值進行處理的訓練模型。藉由非監督式學習，我們可以使用維度縮減的方法，來為一般資料與異常資料個別提供解決方案。我們將在第四章建立一個異常檢測系統。

資料漂移（Data drift）

機器學習模型需要察覺資料的漂移狀況。如果要進行預測判斷的資料，在統計上不同於當時用來訓練的資料，那麼就必須基於目前的資料，對模型進行重新訓練。如果模型並未重新訓練，或者沒意識到資料產生漂移的狀況，那麼，模型在針對目前資料進行預測判斷的品質將會受到影響。

通過使用非監督式學習建立機率分佈，我們可以評估當前資料與訓練資料相異的情況——如果兩者相異程度夠大，我們可以自動觸發再訓練。我們將在第十二章中，探索如何構建這些資料鑑別器。

進一步探索監督式學習

在深入研究非監督式學習之前，讓我們先了解一下監督式學習演算法及其工作原理，這將有助於了解非監督式學習在機器學習生態圈內的定位與擅長主題。

在監督式學習中，存在兩種主要類型的問題：**分類**（*classification*）和**回歸**（*regression*）。在分類問題中，人工智慧必須正確地將待分類項目，分類為一種或多種類別。如果是只有兩種分類的問題，則稱為**二元分類**（*binary classification*）；如果有三個或更多類的問題，則稱為**多元分類**（*multiclass classification*）。

分類問題也稱為**離散**（*discrete*）預測問題，因為每種分類都是一個離散群組。分類問題也可稱為**定性**（*qualitative*）問題或**質性**（*categorical*）問題。

在回歸問題中，人工智慧必須預測**連續**變量而不是離散變量，因此，回歸問題也可稱為**定量**（*quantitative*）問題。

監督式的機器學習演算法可以非常簡單，也可以非常複雜，但目標都是圍繞著給定的資料標籤，進行最小化成本函數或錯誤率（或最大化價值函數）。

碁峰 www.gotop.com.tw

如前所述，我們最關心的是機器學習的解決方案能力在從未見過資料上的泛化能力。監督式學習演算法的選擇，對於最小化泛化誤差是十分重要的。

為了追求盡可能低的泛化誤差，演算法模型的複雜性應該與反應資料樣貌的實際函數複雜性相匹配。我們的確是不知道實際函數是什麼，因為如果我們知道，便不需要使用機器學習來創建一個模型——我們可以直接對這個函數求解，來找到正確的解答。但也由於我們不知道這個實際函數是什麼，我們選擇了一個機器學習演算法來檢驗假設函數，並找到最接近真實函數的模型（即具有最低的泛化誤差）。

如果演算法模型比實際函數更簡單，我們就會低度擬合於資料。在這種情況下，我們可以透過選擇其他可以為更複雜函數進行塑模的演算法來改善泛化誤差。但是，如果演算法設計了一個過於複雜的模型，我們將過度擬合於資料，而導致模型對從未見過資料的預測能力低落，進而提高泛化錯誤。

換句話說，選擇較複雜的演算法並不總是正確的選擇，有時較簡單的演算法反而是更好的選擇。每個演算法都有優點、缺點和假設，基於給定的資料和問題，使用不同的演算法來解決，對掌握機器學習是非常重要的課題。

在本章接下來的部分，我們將描述一些最常見的監督式演算法（包括一些真實的應用案例），然後再接著介紹非監督式演算法[3]。

線性方法

最基本的監督式學習演算法藉由輸入特徵和我們希望進行預測的輸出變量之間的簡單線性關係，進行塑模。

線性回歸（Linear regression）

所有演算法中最簡單的是**線性回歸**，它假設輸入變量（x）和單個輸出變量（y）之間存在線性關係，並以此建立模型。如果實際上，輸入和輸出之間的關係是線性的，且輸入變量之間不是高度相關的（這樣的情況稱為**共線性**（*collinearity*）），線性回歸可能是一個合適的選擇。如果輸入與輸出的關係更複雜或非線性，線性回歸方法將低度擬合於資料[4]。

3 這個列表並非詳盡無遺，但的確列出了最常見的機器學習演算法。

4 可能存在其他使線性回歸成為差勁選擇的潛在問題，包括離群值、相關聯的誤差項和非固定的誤差項的變異值。

碁峰 www.gotop.com.tw

因為它很簡單，所以藉由此演算法建出的關係模型可以非常直截了當地被解讀。**可解釋性**（*interpretability*）是使用機器學習時，一個非常重要的考慮因素，因為解決方案需要被產業內的技術人員與非技術人員理解和實際運用。沒有可解釋性的解決方案會成為無從判讀的黑盒子。

優點

線性回歸很簡單，可被解釋，並且難以過度擬合，因為它不能處理過於複雜的關係。當輸入和輸出變量之間的關係是線性的時候，它是一個十分恰當的選擇。

缺點

當輸入與輸出變量之間的關係不為線性時，線性回歸將低度擬合於資料。

應用

由於人體重量與人體高度之間真正的關係為線性，所以線性回歸對於使用高度作為輸入來預測體重是很好的選擇，反之亦然，使用體重作為輸入變量來預測高度。

邏輯回歸（Logistic regression）

最簡單的分類演算法是**邏輯回歸**，它也是線性的方法，但使用邏輯函數轉換輸出的預測結果。這種轉換的輸出結果是**類別機率**（*class probabilities*）。換句話說，就是待預測實例屬於各類別的機率，而各類別機率的總和為 1。每個待預測實例將屬於具有最高機率的類別。

優點

與線性回歸一樣，邏輯回歸很簡單且可解釋。當我們試著進行預測的類別是不重疊且線性可分的時候，邏輯回歸是很好的一個選擇。

缺點

當所預測的類別不是線性可分的時候，邏輯回歸將無法處理這樣的分類問題。

應用

當分類的類別幾乎不重疊的時候，比方說，兒童的身高對比成人的身高，利用邏輯回歸進行分類，將有不錯的成效。

碁峰 www.gotop.com.tw

基於近鄰方法

另一種非常簡單的演算法是基於近鄰方法。基於近鄰方法的概念都是屬於**惰性學習方法**，因為它們學習如何標記新的資料點是基於新點與現有標記點的近似程度。不像線性回歸或邏輯回歸，基於鄰域的模型並不經由訓練產生一個可以預測新資料點標籤的模型；相反地，這些模型預測新資料點的標籤，是基於新資料點到已經存在且具有標籤的資料點之間的距離。惰性學習（lazy learning）也稱為基於**資料實例學習**（*instance-based learning*）或**非參數方法**（*nonparametric methods*）。

k-nearest neighbors

最常見的基於近鄰方法是 *k-nearest neighbors*（*KNN*）。KNN 藉由查找與新資料點最相近的 k 個已經具有標籤的資料點（k 是整數值），並且讓這些已經具有標籤的近鄰點投票如何標記新的資料點。預設情況下，KNN 使用歐氏距離來測量最接近的距離。

k 值的選擇非常重要。如果將 k 值設得非常低，KNN 在標籤揀選上將變得非常寬鬆，畫出十分貼合於資料分佈的邊界線，因此，可能過度擬合於資料。如果是 k 值設得非常高，KNN 的標籤揀選會變得過於嚴格，繪製過於僵固的邊界，進而可能低度擬合於資料。

優點

與線性方法不同，KNN 非常靈活，擅長學習更複雜以及非線性的關係。即便如此，KNN 仍然簡單且可被解釋。

缺點

隨著資料與特徵的數量增加時，KNN 的效果並不好。在資料密集且高維的空間內，KNN 的計算效率低下，因為它需要計算從新的資料點到附近許多已具有標籤資料點的距離，以便預測新資料點的標籤。KNN 不能藉由將參數減少到剛好能完成預測的數量，來得到一個高效的模型。此外，KNN 對 k 值的選擇非常敏感。當 k 設置得太低時，KNN 會過度擬合，而當 k 值設得太高時，KNN 會低度擬合。

應用

KNN 經常用於推薦系統，例如針對電影（Netflix）、音樂（Spotify）、朋友（Facebook）、照片（Instagram）、搜索（谷歌）和購物（亞馬遜）進行喜好預測。例如，KNN 可以根據類似用戶的喜好（稱為**協同過濾**（*collaborative filtering*））或用戶過去喜歡的內容（稱為**基於內容的過濾**），預測用戶會喜歡什麼。

碁峰 www.gotop.com.tw

基於樹的方法

不採用線性方法，取而代之的是我們可以藉由已知的標籤讓模型方法構建一個**決策樹**（*decision tree*），所有資料實例被**分割**或者是被**分層**為許多區塊。當分割完畢後，每個區域對應於一個特定的標籤（用於分類問題）或一個範圍的預測值（用於回歸問題）。此過程類似讓人工智慧基於做出更好決策或預測的明確目標，自動建立規則。

單一決策樹

最簡單的基於樹的方法是**單一決策樹**（*single decision tree*），透過遍歷所有的訓練資料與既有的標籤，建立用於分割資料的規則，並使用此樹針對驗證集或者是測試集等未見過的資料進行預測。但是單一決策樹通常很難延伸它基於訓練資料所學到的知識到未見過的資料上，因為它通常會在僅只一次的訓練迭代中，過度擬合於訓練資料。

裝袋法（Bagging）

為了改進單一決策樹，我們可以引入**自助聚合**（*bootstrap aggregation*）（以**裝袋法**（*bagging*）為人所熟知）。我們從訓練資料中，使用**多次隨機實例採樣**（*multiple random samples of instances*），並且為每次採樣建立一個決策樹，然後對每個資料實例進行預測，預測方式為透過平均每個樹的預測結果來作為每個資料實例的預測結果。藉由採樣**隨機化**（*randomization*）和多個樹的平均結果，這樣的方法也被稱為**整體方法**（*ensemble method*）。裝袋法可以解決一些單一決策樹帶來的資料過度擬合問題。

隨機森林（Random forests）

我們不僅可以透過對資料實例進行採樣，還可以透過對待預測的標籤進行採樣，來進一步改善過度擬合。對於隨機森林，我們不僅如同裝袋法（bagging）從訓練用的資料實例進行多次的隨機採樣外，也對用於每棵決策樹中分支條件的**待預測標籤**進行**隨機採樣**，而非使用所有的待預測標籤。用於分支條件的待預測標籤數量通常是使用待預測標籤總數量的平方根作為數量基準。

透過對待預測標籤進行採樣，隨機森林演算法可以建出彼此相關性更低的決策樹（與裝袋法中的決策樹相比），進而減少過度擬合和改善泛化誤差。

碁峰 www.gotop.com.tw

提升方法（Boosting）

另一種基於樹的方法稱作**提升方法**。如同裝袋法一般，這種演算法也會建出多棵樹，但它是**依序地建出每一棵決策樹**，利用前棵決策樹所學得的資訊，來改善下一棵決策樹的預測結果。提升方法中每棵樹的深度，都保持地十分淺，僅含有一些決策分支點，按著逐棵的決策樹，進行緩慢地學習。在所有基於樹的方法中，*gradient boosting machines* 的表現最好，並且是諸多機器學習比賽中的常勝軍[5]。

優點

基於樹的方法是在預測問題任務上，表現最好的監督式學習演算法之一。這些方法能夠捕捉資料中複雜的關係是透過一次學習一個規則的方式，學習到許多簡單的規則來達成。它們也能處理資料缺失與特徵分類的問題。

弱點

基於樹的方法的可解釋性很差，尤其是在需要許多規則，才能做出好的預測的情況下。運算效能隨著特徵增加也會成為一個問題。

應用

對於預測問題，gradient boosting 和隨機森林的表現都非常的優良。

支持向量機（Support Vector Machines）

我們可以使用演算法和已知的資料標籤在空間中建構超平面來分類資料，而不是構建樹來進行資料分類。這方法稱為**支持向量機**（*support vector machines, SVM*）。SVM 允許一些違反這種分類原則的行為，也就是說，並非超空間區域內的所有點都需要具有相同的標籤，但是某個標籤的邊界定義點與另一個標籤的邊界定義點的距離，應盡可能地最大化。此外，邊界不必是線性的，我們可以使用非線性內核（kernel）靈活地分類資料。

類神經網路

我們可以藉由單層輸入層、多層隱藏層以及單層輸出層所組成的類神經網路，進行資料特徵的學習[6]。輸入層使用特徵值作為輸入，輸出層的輸出值試著逼近預期的應變數。隱

5 更多有關 gradient boosting 在機器學習競賽的資訊，請參閱 Ben Gorman 的博客文章（*http://bit.ly/2S1C8Qy*）。

6 更多有關神經網路的資訊，請看由 Ian Goodfellow、Yoshua Bengio 和 Aaron Courville（麻省理工學院出版社）撰寫的《深度學習》（*Deep learning*）（*http://www.deeplearningbook.org/*）。

碁峰 www.gotop.com.tw

藏的圖層是一個巢狀階層結構的概念，每一層（或概念）都試圖理解前一層如何關聯於輸出層。

神經網路使用這種階層概念，從簡單到複雜一層一層地逐步學習複雜的概念。神經網路是逼近目標函數強而有力的方法之一，但容易過度擬合且很難被妥善解釋，我們將在本書後面詳細探討它的缺點。

進一步探索非監督式學習

我們現在將注意力轉向沒有標籤的問題。非監督式學習演算法會嘗試學習資料的基礎結構，而不是嚐試作出預測。

維度縮減

維度縮減演算法（*dimensionality reduction algorithm*）是演算法中的一個分類，它將原始的高維度輸入資料映射到一個低維度空間，同時過濾掉了與整體資料不那麼相關的特徵，並且盡可能多地保留資料中令人感興趣的特徵。

維度縮減讓非監督式學習能更正確地識別樣式（patterns）並更有效率地解決大規模所導致的昂貴運算問題（通常涉及圖像、影片、語音和文字）。

線性投影

關於維度縮減有兩個主要分支，一個是線性投影，另一個則是非線性維度縮減。我們首先從線性投影開始介紹。

主成分分析（Principal component analysis, PCA） 是一種學習資料基礎結構的方法，用於識別整個特徵集合中哪些特徵最為重要，且能解釋資料實例之間的可變性。並非所有特徵都同等重要，有些特徵在資料中的值變化不大，因此，這些特徵在解釋資料集時就沒那麼有用，而另外其他特徵的值在資料集中可能有明顯的變化，並且值得仔細探索，因為它們會幫助我們設計的模型更好地分類資料。

在 PCA 中，演算法找到低維度空間表示資料的方法，同時盡可能的最大化地保持資料之間的變異程度。我們留下的維度數量明顯少於原始資料的總維度數量（即總特徵的數量）。由於轉移至這個低維度的空間，我們會損失一些資料中的變異，但是這將使得資料內的結構更容易被識別，並且讓我們在執行如分群的任務時更加地有效率。

碁峰 www.gotop.com.tw

PCA 有數種變形，我們稍後將在本書進行探討。這些變化包括 mini-batch 變形（如**增量式** *PCA*（*incremental PCA*））、非線性變形（如 *kernel PCA*），以及稀疏變形（如 *sparse PCA*）。

奇異值分解（Singular value decomposition, SVD） 學習資料基礎結構的另一種方法是降低原來特徵所組成的矩陣的秩（rank），使得原來的矩陣可以使用擁有較小的秩的矩陣所組成的線性組合來表示。這被稱為 *SVD*。為了生成有較小的秩的矩陣，SVD 保持擁有最多資訊的原始矩陣的向量（即最大的奇異值）。這個有較小的秩的矩陣擷取原始特徵空間中最重要的特徵。

隨機投影（Random projection） 一個類似於維度縮減的演算法，它將點從較高維的空間投影至較低維的空間，但同時保持點和點之間的距離。我們可以使用**隨機高斯矩陣**（*random Gaussian matrix*）或**隨機稀疏矩陣**（*random sparse matrix*）來實現這個目標。

流形學習（Manifold learning）

PCA 和隨機投影都是將資料從高維度空間線性投影到低維度空間。替代使用線性投影，**流形學習**或稱**非線性維度縮減**可以更好地進行非線性轉換。

Isomap *Isomap* 是流形學習的一種。這種演算法會透過估算每個點與鄰近點的**捷線**（*geodesic*）或**曲線距離**（*curved distance*），而非使用歐式距離（Euclidean distance）來學習資料流形的內蘊幾何。Isomap 利用這樣的估算將原始高維度的空間嵌入一個低維度的空間。

t-distributed stochastic neighbor embedding（t-SNE） 另一個維度縮減的方法稱為 *t-SNE*，它將高維度空間的資料嵌入至只有二維或三維的空間，這樣的空間可以讓轉換的資料被視覺化的呈現出來。在二維或者是三維的空間裡，相似的資料會被聚集在一起，反之則被分離開來。

字典學習（Dictionary learning） **字典學習**是一種為人所知的方法，它會學習資料基礎結構的稀疏表示方式。這些具代表性的要素是簡單的、以二進制向量（0 和 1）構成，並且資料集中的每個實例都可以藉由這些代表性要素的加權總和，進行重建。這個由非監督式學習所產生的矩陣（稱作**字典**）大都由零值組成，僅有少量元素為非零值。

碁峰 www.gotop.com.tw

透過建立這樣的字典，這個演算法能夠有效地識別原始特徵空間中，最顯著且具代表性的要素，也就是那些擁有最多非零權重的元素。較不重要的具代表性要素中，則只有很少的非零權重。與 PCA 一樣，字典學習非常適合學習資料的基礎結構，這將有助於分類資料和識別有趣的樣式。

獨立成分分析（Independent component analysis）

無標籤資料一個常見的問題是有許多各自獨立的信號被一起嵌入混合在被給定的特徵中。使用**獨立成分分析**（*ICA*），我們可以將這些混合的信號分離成各自獨立的要素。分離完成後，我們可以透過將我們生成的各個要素，以某種組合方式加在一起來重建原始的特徵。ICA 通常用於信號處理任務（例如，用於在繁忙咖啡館的音檔中，識別各個獨立的聲音）。

隱含 Dirichlet 配置（Latent Dirichlet allocation）

非監督式學習也可以透過學習來解釋一個資料集中，為什麼某些部分會彼此相似。這需要了解資料集內部未被觀察到的元素。有一種被稱為**隱含 *Dirichlet* 配置**（*LDA*）的方法，可以為此提出解法。例如，考慮一個包含許多單字的純文字文件。文件中的這些單字並非純粹隨機地排置，相反地，它們展現出一些結構。

可以將這個結構塑模成為未被觀察元素，這些元素稱作主題（topics）。在訓練後，LDA 能夠解釋一個有少量主題的文檔，其中每個主題都有一小組經常使用的單字。這就是 LDA 能夠捕捉的隱藏結構，這個結構可以幫助我們更好地解釋先前文件中的非結構化文體。

維度縮減將原始特徵集大小減小成另一個較小的特徵集合，但該集合保有原始特徵集合中最重要的部分。基於此，我們可以在這個較小的特徵集上執行其他的非監督式演算法，來找尋資料內令人感興趣的樣式，抑或在我們有標籤資訊下，透過使用較小的特徵矩陣，而不是使用原始的特徵矩陣，來加速監督式演算法的訓練速度。

碁峰 www.gotop.com.tw