前言

自從 2017 年，我在史丹佛大學首執機器學習課程教鞭，許多人向我諮詢如何在他們的組織中部署 ML 模型。這些問題可以是概括性的，例如：「我應該使用什麼模型？」、「我應該多久重新訓練一次我的模型？」、「如何檢測資料分布變化？」、「如何確保訓練期間使用的特徵與推理期間使用的特徵一致？」

這些問題也可以是具體性的，例如：「我相信從批量預測切換到在線預測將使我們的模型效能得到提升，但我如何說服經理讓我這樣做？」或者「我是公司最資深的資料科學家，最近的任務是建立第一個機器學習平台。我要從何開始？」

對於這些問題，我的簡短回應總是：「視情況而定。」而長篇回答通常涉及數小時的討論，以了解提問的來龍去脈、提問者實際想達成的目標、以及針對特定用例施行不同方法之優劣。

ML 系統既複雜又獨特。它們很複雜，因為它們由許多不同的組件（ML 演算法、資料、業務邏輯、評估指標、底層基礎設施等）組成，並涉及許多不同的利益相關者（資料科學家、機器學習工程師、商業領袖、用戶，還可能推展至社會層面）。ML 系統是獨一無二的，因為它們依賴於資料，而不同用例的資料又大有不同。

例如，兩家公司可能在同一個領域（電子商務）並且有相同的問題，他們希望以 ML 解決（推薦系統），但結果兩家公司的機器學習系統可能有著不同模型架構，使用不同特徵集，根據不同指標進行評估，並帶來不同的投資回報。

許多關於 ML 生產的部落格文章和教程都專注於回答一個特定問題。雖然聚焦討論有助於理解核心觀念，但這會給人一種「可以把這些問題單獨考慮」的印象。實際上，一個組件的更改可能會影響其他組件。因此，在嘗試做出任何設計決策時，有必要將系統作為一個整體來考慮。

本書對 ML 系統進行整體性的探討。它考慮了系統的不同組成部分，以及所涉利益相關者們的目標。本書以實際案例研究進行說明，其中許多是我親自參與的，有大量參考資料支持，並由學術界和業界的 ML 從業人員審閱。對於涉及深入了解某個主題的部分（例如：批量處理與串流處理、儲存和運算的基礎設施，以及負責任的人工智慧）則由專注於該主題的專家進一步審閱。換句話說，本書試圖對上述問題以及其他方面給出細緻入微的答案。

當我第一次為本書奠定基礎講義時，我的想法是：本書是為了我的學生撰寫，好讓他們未來作為資料科學家和 ML 工程師做好準備，以達相關職能需求。然而我很快意識到，我在這個過程中也學到了很多東西。與早期讀者分享的初稿，引發了許多對話，這些對話檢驗了我的假設，迫使我考慮不同的觀點，並引領我探知新的問題和方法。

既然這本書在你手裡，我希望這個學習過程能夠持續下去，因為你擁有自己獨特的經歷和觀點。若你對本書有任何回饋，請隨時透過以下方式與我分享：我負責運作的 MLOps Discord 服務器（*https://discord.gg/Mw77HPrgjF*）（你也可以在其中找到本書的其他讀者）、Twitter（*https://twitter.com/chipro*）、LinkedIn（*https://www.linkedin.com/in/chiphuyen*），或你可以在我的網站（*https://huyenchip.com*）上找到其他聯絡渠道。

本書適用對象

本書適用於任何想利用 ML 解決現實世界問題的人。書中提及的「ML」指的是深度學習和經典演算法，傾向於大規模 ML 系統，例如在中大型企業和快速成長型新創公司中看到的系統。規模較小的系統往往不那麼複雜，書中列出的綜合方法對此類系統的幫助可能較少。

因為我的背景是工程學，所以本書的語言面向工程師，包括 ML 工程師、資料科學家、資料工程師、ML 平台工程師和工程經理。你可能會遇到以下情況之一：

- 收到一個業務問題和一大堆原始資料。你想進行資料工程，並選擇正確的指標來解決此問題。
- 初始模型在離線實驗中效能良好，你希望部署它們。
- 部署後，幾乎沒有模型效能的反饋，你想找到一種方法來快速檢測、糾錯和解決模型在生產環境可能遇到的任何問題。
- 為團隊開發、評估、部署和更新模型的過程中，大多是手動、緩慢並且容易出錯的。你想自動化並改善此過程。
- 組織中的每個 ML 用例都各自使用自己的工作流程進行部署，你希望奠定跨用例共享和重用的基礎（例如：模型儲存庫、特徵儲存庫、監控工具）。
- 你擔心機器學習系統可能存在偏差，並希望這是負責任的系統！

如果你屬於以下群體之一，你也可以從本書中受益：

- 工具開發人員，想識別生產環境 ML 生態中服務不足的領域，希望弄清楚如何在生態系統中定位你的工具。
- 在行業中尋找 ML 相關職位的人。
- 正考慮採用 ML 解決方案以改進產品和 / 或業務流程的技術和業務領導者。沒有深厚技術背景的讀者可能從第 1、2 和 11 章獲益最多。

本書非入門教材

本書並非旨在介紹 ML。有很多關於 ML 理論的書籍、課程和資源，因此本書避開這些概念，而專注於 ML 實踐。具體來說，本書假定讀者對以下主題有基本了解：

- *ML 模型*，例如聚類、邏輯回歸、決策樹、協同過濾，以及各種神經網路架構，包括前饋、遞歸、卷積和 Transformer
- *ML 技術*，例如監督與非監督、梯度下降、目標函式 / 損失函式、正則化、泛化和超參數調整
- 指標如準確度、F1、精確度、取回率、ROC、均方誤差和對數似然

- *統計概念*如標準差、機率和常態 / 長尾分布等
- *常見 ML 任務*如建立語言模型、異常檢測、物體分類和機器翻譯等

你不必對這些主題瞭若指掌（對於要那些需要費力才能記住其確切定義的概念，例如 F1 分數，我們提供了簡短的註釋作為參考）但你應該對它們的涵義有粗略了解。

雖然本書提到了當前的工具，來說明某些概念和解決方案，但它不是一本教程書。技術隨著時間的推移而發展。工具更新換代很快，但解決問題的基本方法應該更持久。本書為你提供一個框架，來評估最適合用例的工具。當你想使用某個工具時，通常可以直接在網上找到它的教程。因此，本書的程式碼片段很少，而側重於提供大量關於權衡、利弊和具體示例的討論。

瀏覽本書

本書的章節排列方式映射了資料科學家在 ML 專案週期的進程，以及其中可能遇到的問題。前兩章為 ML 專案的成功奠定基礎，我們將從最基本的問題開始：你的專案需要 ML 嗎？它還包括如何為專案選擇目標，以及如何用更簡單的解決方案來構建問題。如果你已經熟悉這些注意事項，並且急於獲得技術解決方案，請跳過前兩章。

第 4 章到第 6 章涵蓋了 ML 專案部署之前的階段：從創建訓練資料、特徵工程、直到在開發環境中開發和評估模型。這是特別需要 ML 和問題領域專業知識的階段。

第 7 章到第 9 章介紹 ML 專案的部署，和部署後的階段。部署模型結束不等於部署過程的結束，許多讀者可能對此有同感，我們將透過一個故事了解這個句話的意思。部署的模型將需要受到監控並時常更新，以適應不斷變化的環境和業務需求。

第 3 章和第 10 章重點介紹 ML 系統所需的基礎設施，讓來自不同背景的利益相關者能夠協同工作，圓滿交付過程。第 3 章側重於資料系統，而第 10 章側重於運算基礎設施和 ML 平台。對此，我們花了很多時間爭論資料系統的探討該深入到什麼程度、又該放在書中哪個段落。資料系統包括資料庫、資料格式、資料移動和資料處理引擎，這些在 ML 課程中往往很少提及，因此許多資料科學家可

能認為它們是低級或無關緊要的。諮詢許多同事後，我肯定了「ML 系統依賴資料」的重要性，決定儘早介紹資料系統的基礎知識，這將有助我們在本書的餘下部分討論與資料相關的問題。

雖然本書涵蓋了 ML 系統的許多技術層面，但 ML 系統始終由人構建、為人構建，並且會對許多人的生活產生巨大影響。要是我寫了一本關於 ML 生產環境的書，卻沒有一章講述人類相關方面，那肯定是我的失職。這是第 11 章，也是最後一章的重點。

請注意，「資料科學家」是一個在過去幾年中發生了很大變化的角色，並且已經有很多討論來確定這個角色應該承擔什麼——我們將在第 10 章探討其中一些觀點。在本書，我們使用「資料科學家」作為總稱，包括從事開發和部署 ML 模型的任何人，其職位可能是 ML 工程師、資料工程師、資料分析師等等人員。

GitHub 儲存庫和社區

本書附有一個 GitHub 儲存庫（*https://oreil.ly/designing-machine-learning-systems-code*），其中包含：

- 基本 ML 概念回顧
- 本書中使用的參考文獻列表，其他進階或更新的資源
- 本書中使用的程式碼片段
- 解決工作流程中可能遇到特定問題的工具列表

我還設置了一個名為 MLOps 的 Discord 服務器（*https://discord.gg/Mw77HPrgjF*），希望你參與討論，並提出有關本書的問題。

本書編排方式

以下是本書使用的字體規則：

楷體字（*Italic*）

代表新術語、URL、電子郵件地址、檔案名稱及副檔名。

第一章

機器學習系統概覽

2016 年 11 月，Google 宣布將多語言神經機器翻譯系統（multilingual neural machine translation system）整合到旗下的 Google 翻譯（Google Translate）服務。Google 表示，這次更新將大幅提升翻譯品質，改進幅度比過去十年所有更新加起來還要大。這是大規模應用人工智能深度神經元網路（artificial neural networks）的首批成功案例之一[1]。

此事重新燃起各界對機器學習（Machine Learning，簡稱 ML）在大規模應用的興趣。更多企業打算以機器學習取代傳統技術方案來解決最具挑戰性的難題。短短五年間，以機器學習為核心的科技已迅速滲透到生活各個範疇，改變了我們獲取資訊、溝通、工作、甚至尋覓愛情的方式，很難想像沒有機器學習科技的生活是什麼樣子。雖說 ML 已獲廣泛應用，其技術在醫療保健、交通運輸、農業、宇宙研究[2]等領域，尚有大量潛在用途等待我們去探索。

很多人一聽見 ML 系統，只會想到機器學習算法（algorithm），像是邏輯斯迴歸模型（logistic regression），或是各種神經網路架構（neural networks）。其實這些算法只佔 ML 系統實際運作中很小的部分。ML 專案始於業務需求，還包括用

1 Mike Schuster，Melvin Johnson 及Nikhil Thorat《Zero-Shot Translation with Google's Multilingual Neural Machine Translation System》，*Google AI Blog*，2016 年 11 月 22 日，*https://oreil.ly/2R1CB*。

2 Larry Hardesty，《A Method to Image Black Holes》，*MIT News*，2016 年 6 月 6 日，*https://oreil.ly/HpL2F*。

戶與開發人員介面、數據技術架構、模型的建立和管理邏輯，以及支持相關邏輯的基礎運算架構。圖 1-1 展示了 ML 系統的組成部分，和本書各章所涵蓋之處。

MLOps 與ML 系統設計的關係

「MLOps」中的「Ops」來自「DevOps」一詞。「DevOps」全稱「Developments and Operations」，即「開發與運維」。所謂運維化（operationalize），即涉及部署、監控、維護等作業，MLOps 就括了把 ML 系統搬到實際運作環境的一系列最佳作業流程和工具。

至於 ML 系統設計（ML systems design），則以系統思維看待 MLOps，綜觀全局，確保利益相關者充分合作，滿足各方目標和需求。

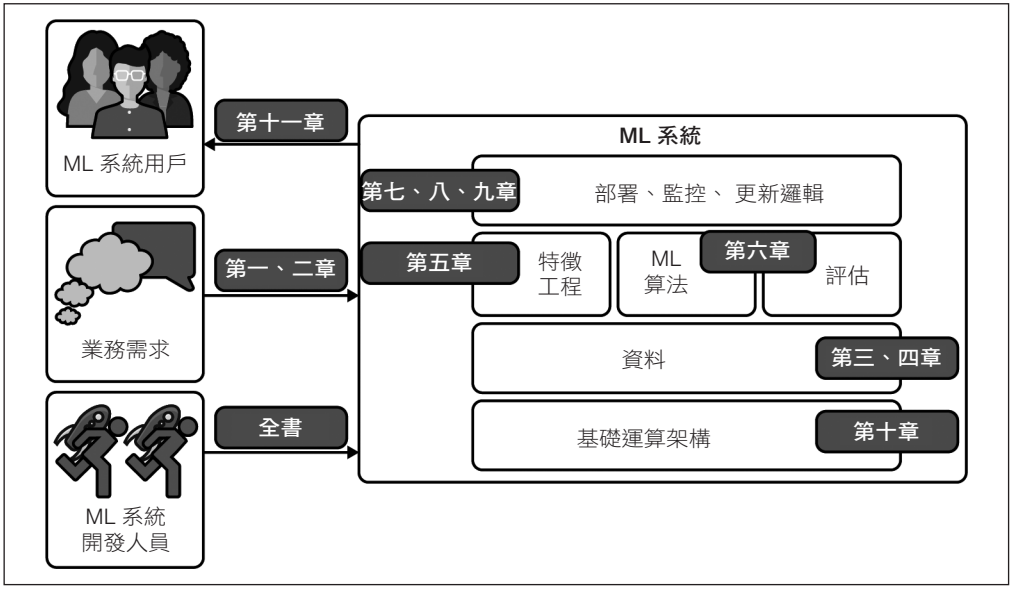

圖 1-1　ML 系統各部分。提到機器學習，人們一般只會想到起「ML 算法」，然而這只是 ML 系統的一小部分。

市面上不乏探討 ML 算法的芸芸好書。本書並不著眼於個別算法的細節，而是側重於協助讀者全面理解 ML 系統。換言之，不論哪種算法，讀者都可以按照本書提供的框架開發最佳的 ML 解決方案。即使演算法頻頻更迭，這套框架也可以適用。

本章旨在概述機器學習模型在實際環境運行的所需條件。探討 ML 系統開發前，應先仔細考慮使用案例是否適合套用 ML。關於這一點，我們會分享一些常見案例。接著我們會檢視 ML 在實際應用和學術研究領域的分別、還有 ML 系統與傳統軟體架構相異之處，來說明部署 ML 系統的挑戰。經常參與應用 ML 系統開發的讀者，對本章內容或許已有一定了解；但如果你的機器學習知識只集中在學術研究領域，本章將助你一窺 ML 實作之全貌，為開發首個成功應用系統打下基礎。

使用機器學習的時機

機器學習方案在業界的應用快速增長，證實 ML 的確有效解決多個範疇的問題，發展進程振奮各界，也帶來一些炒作。我們需要明白，ML 並非解決所有問題的「神器」。即使 ML 方案能解決問題，ML 也不一定就是最佳解方。啟動 ML 專案前，我們應先考慮兩點——ML 是否必需？ ML 是否符合成本效益[3]？

要了解 ML 能做什麼，請先閱讀以下概述：

> 機器學習是指從（3）既有資料（1）學習（2）複雜規律模式，並利用習得模式（4）預測（5）未見資料的方法。

針對以上楷體字詞之涵義，可以進一步了解 ML 在解決什麼：

1. *學習：系統具備學習能力*

   關聯式資料庫（relational database）缺乏學習能力，因此不屬於 ML 系統。在關聯式資料庫內，你只能表明兩列的關係，但資料庫本身不能了解兩列的實際關係。

   要讓 ML 系統學習，就要有學習對象。大部分情況下，ML 系統透過資料學習。在監督式學習領域，ML 學習成對的輸入 / 輸出範例，然後基於任意未知資料，給出結果。假設你希望建立一個 ML 系統，預測 Airbnb 的房租價格，你需要提供一個資料集，輸入範例包括房源的特徵資料（平方英尺、房間數目、街區、提供的必需品、評價等），相對的輸出範例則為租金資料。訓練學習後，只需向 ML 系統提供新房源的特徵資料，便能評估其租金。

3 我沒有提及「有了 ML 是否就足以解決」這一點，因為答案是總是否定的。

2. 複雜規律模式：學習旨在了解箇中模式，模式複雜才值得學習

只有當規律模式（pattern）確實存在，ML 方案才有用武之地。正常人不會花錢開發預測投擲公正骰子結果的 ML 系統，因為這些結果的產生方式沒有規律可言[4]。至於股票的定價，卻涉及一定規律，很多公司投資數十億美元開發 ML 系統，就是希望透過 ML 學習，了解這些規律。

即使有了算法和資料集，也未必能夠捕捉現存規律模式。就好像要了解馬斯克（Elon Muck）的貼文如何影響加密貨幣價格，箇中或有規律模式，但我們還需要花很大心思反覆訓練和評估模型，才能驗證其存在。就算是訓練後的模型不能合理預測加密貨幣價格，也不意味著規律模式不存在。

模式複雜才值得學習。比方說，在 Airbnb 這樣的網站上有許多住宿房源，你希望按美國州份分類房源地點。只需根據房源的郵區編號（ZIP code），你就可查找出地點所在的州份。這樣的規律模式簡單得很，一個尋找表（lookup table）就可以解決問題，不需要 ML 系統。

但房源的特徵資料與房租的關係，就很難單靠手動操作實現規律模式。這種情況下，ML 是個好選擇。我們不用告訴 ML 系統如何根據地點特徵計算租金，只需把租金和地點特徵導入 ML 系統，系統即可自行學習各參數之間的模式，是故 ML 又名軟體 2.0[5]。圖 1-2 展示 ML 方案與一般軟體方案之異同。

ML 在一些需要了解複雜規律模式的任務，比如物件探測（object detection）和語音辨識（speech recognition）上的效果十分令人滿意。在此需說明一點，「複雜」一詞是針對機器而言。一些人類覺得困難的工作，對機器來說相對容易，比方說計算某數字的 10 次方；但很多我們覺得簡單的事情，像是判斷照片中是否有貓，從機器的角度來看，就是困難的任務。

4 規律模式（pattern）跟機率分布（distribution）的概念不同。我們知道擲骰子結果依循機率分布，但結果生成的方式是不規律的。

5 Andrej Karpathy，《Software 2.0》，*Medium*，2017 年11 月11日，*https://oreil.ly/yHZrE*。

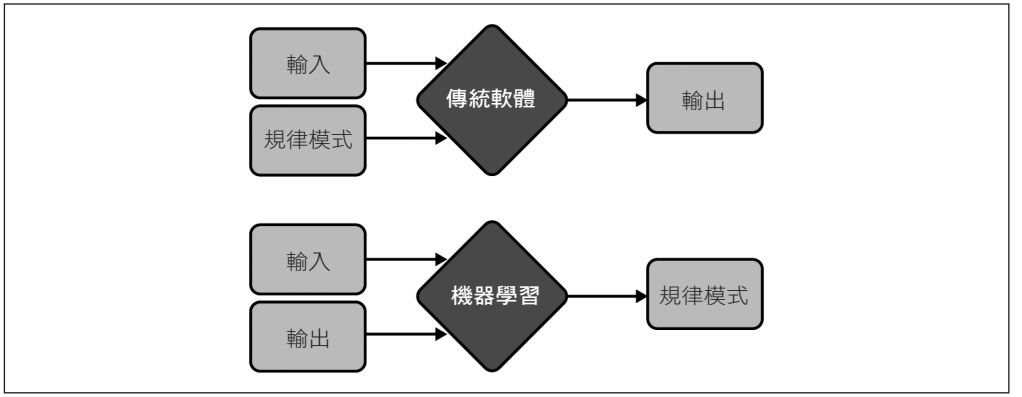

圖 1-2　ML 方案以輸入和輸出來了解箇中模式，而非透過輸入和模式來計算輸出。

3. 現存資料：要嘛是現有資料，要嘛可蒐集資料

既然 ML 系統是藉由學習資料來得出規律模式，我們就需要確保資料的供給。建立模型來預測一個人的年繳稅款，看似是個有趣的項目，但除非能夠訪問大量人口稅收和收入資料，否則是不可能建立這個模型的。

零樣本學習（zero-shot learning）（*https://oreil.ly/ZshSg*）（亦稱零資料學習，zero-data learning）能夠在沒有相關資料的情況下，讓 ML 系統給出良好的預測結果。然而，這種 ML 系統往往曾進行過另外的任務資料訓練，而這些任務通常與進行零樣本學習的任務相關。因此，即使不需要資料來學習手頭上的任務，從根本上來看，系統仍然依賴資料。

我們也可以在缺乏資料的情況下啟動 ML 系統。持續學習（continual learning）讓我們可以在沒有任何資料的情況下先部署 ML 模型，系統實際運作時，新資料便持續傳入模型以供學習[6]。但需留意訓練不足的模型會帶來一定風險，例如糟糕的客戶體驗。

要是沒有資料，又沒有持續學習架構，許多公司則會先推出提供預測的產品，後端沒有 ML，單靠人手操作。這種佯裝的套路還是能夠生成預測資料，假以時日，就可以利用這些資料來訓練 ML 模型了。

6　線上學習（online learning）將在第九章說明。

4. 預測：這是一個涉及預測的問題

ML 模型旨在進行預測，因此它們只能解決需要預測答案的問題。如果問題能夠受惠於大量低成本而接近現實的預測，ML 可能特別有吸引力。在英語中，「預測」（predict）指「估計未來的價值」。比如，明天的天氣怎麼樣？誰將贏得今年的超級碗？用戶接下來想看什麼電影？

隨著預測機器（例如 ML 模型）變得越來越有效，越來越多問題被重新定義為預測問題。無論你有什麼問題，你都可以將其界定為：「這個問題的答案是什麼？」問題可以是關於未來，現在，甚至過去的事情。

運算密集型問題就是將問題重新界定為預測性問題中一個非常成功的例子。要運算一個過程的確切結果，可能比 ML 的計算成本更高、更耗時。與其運算確切結果，你可以將問題定義為：「這個過程的結果會是什麼樣子？」並使用 ML 模型生成近似答案。模型輸出將是實際輸出的近似值，但通常已經夠好了。你可以在圖形渲染中看到很多用例，例如：圖像去雜訊和屏幕空間著色[7]。

5. 未見的資料：未見的資料與訓練資料有著同樣的規律模式

你的模型從現有資料學習到的規律模式，只有在未見的資料有著同樣規律模式時才有用。如果一個預測應用程式在 2020 年聖誕節會否被下載的模型，使用了 2008 年的資料進行訓練，那麼該模型的效能將不會很好，當時 App Store 上最受歡迎的應用程式是 Koi Pond。Koi Pond 是什麼東西？就是嘛。

用術語來說，這意味著未見的資料和訓練資料應該來自相似的分布。你可能會問：「如果資料是未見的，我們怎麼知道它來自什麼分布？」我們確實不知道，但我們可以做出假設（比如我們可以假設用戶明天的行為與今天的用戶行為不會有太大差異）並希望我們的假設成立。如果這個假設不成立，我們將得到一個效能不佳的模型，我們可以透過監控（如第 8 章所述）以及在生產環境中進行測試（如第 9 章所述）來發現這一點。

---

7 Steke Bako、Thijs Vogels、Brian McWilliams、Mark Meyer、Jan Novák、Alex Harvill、Pradeep Sen、Tony Derose 和 Fabrice Rousselle，《Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings》，*ACM Transactions on Graphics* 36，no. 4 (2017): 97, *https://oreil.ly/EeI3j*；Oliver Nalbach、Elena Arabadzhiyska、Dushyant Mehta、Hans-Peter Seidel 和 Tobias Ritschel，《Deep Shading: Convolutional Neural Networks for Screen-Space Shading》，*arXiv*，2016 年，*https://oreil.ly/dSspz*。

基於當今大多數 ML 算法的學習方式，如果你的問題具有以下額外特徵，ML 解決方案將特別出色：

6. 問題是重複的

   人類非常擅長小樣本學習：你可以給孩子們看幾張貓的照片，他們中的大多數人下次看到貓時就會認出這是一隻貓。儘管小樣本學習研究取得了令人振奮的進展，但大多數 ML 算法仍需許多範例來學習規律模式。當一項任務是重複性的，每個模式都會重複多次，這使得機器更容易學習它。

7. 錯誤預測的代價很低

   除非 ML 模型的效能一直是 100%（這對於任何有意義的任務來說都是極不可能的），否則模型就會出錯。當錯誤預測的成本很低時，ML 尤其適用。例如當今 ML 最大的用例之一是推薦系統，因為即使推薦了糟糕的項目， 這種錯誤通常是可以原諒的——用戶只是不會點擊它。

   如果一個預測錯誤可能導致災難性後果，但平均而言，正確預測的好處超過錯誤預測的成本，ML 可能仍然是一種合適的解決方案。開發自動駕駛汽車具有挑戰性，因為算法錯誤可能導致死亡。然而，許多公司仍然希望開發自動駕駛汽車，因為在統計學的角度，只要自動駕駛比真人駕駛更安全，就有可能挽救許多生命。

8. 具一定規模

   ML 解決方案通常需要對資料、運算、基礎設施和人才進行大量的前期投資。這些解決方案若能被大量的應用在案件中，才是有意義的。

   「具一定規模」對於不同的任務意味著不同的事情，總而言之，它意味著做出大量預測。包括每年對數百萬封電子郵件進行分類，或者每天預測應該將數千個支援個案轉發到哪些部門。

   一個問題可能看起來是單一的預測，但它實際上是一系列的預測。例如：一個預測誰將贏得美國總統大選的模型，似乎每四年才做出一次預測，但實際上它可能以每小時甚至更高的頻率做出預測，因為該預測必須不斷更新，以納入新的資訊。

   具一定規模的問題也意味著需要蒐集大量資料，這對於訓練 ML 模型很有用。

9. 規律模式不斷變化

文化在變、口味在變、科技在變。今天流行的東西，明天可能就是舊聞。以垃圾郵件分類的任務為例，今天看出垃圾郵件的端倪可能是「一位尼日利亞王子」，但明天可能是「一位心煩意亂的越南作家」。

如果你的問題涉及一種或多種不斷變化的規律模式，則寫死方案（hardcoded solutions，例如定下硬性規則）可能很快就會過時。要弄清問題發生了怎樣的變化，才相應更新硬性規則，這可能過於昂貴，甚至不可能。因為 ML 從資料中學習，所以你可以使用新資料更新 ML 模型，而不必弄清楚資料是如何變化的。你也可以設置系統來適應不斷變化的資料分布，我們將在第 264 頁「持續學習」中探討。

用例可以繼續增加，而隨著 ML 採用在行業變得成熟，列表只會變得更長。儘管 ML 可以很好地解決一部分問題，但它不能也 / 或不應該用於解決的問題也有很多。當今大多數 ML 算法不應在以下任何情況下使用：

- 不符合道德倫理的情況。我們將在第 343 頁的「案例研究 I：自動評級的偏誤」探討一個案例，在該案例中，使用 ML 算法可能是不符合道德倫理的。
- 使用更簡單的解決方案就可以搞定。在第 6 章，我們將介紹 ML 模型開發的四個階段，第一階段就是非 ML 解決方案。
- 這不符合成本效益。

然而，即使 ML 不能解決你的問題，也可以將你的問題分解成更小的部分，並使用 ML 來解決其中的一些問題。例如，如果你無法構建一個能回答客戶所有問題聊天機器人，可以先構建一個 ML 模型來預測客戶的問題是否與常見問題相符。如果相符，則引導客戶找到答案。如果不相符，則引導客戶聯繫客服。

我還想告誡一點：不要因為新技術的成本效益比現有技術低就放棄它。大多數技術進步都是漸進的。一種技術現在可能效率不高，但隨著時間推移，隨著投資增加，它的效率可能會提升。如果你等到這項技術向同業顯示其價值後再進入，最終可能落後競爭對手數年甚至數十年。

機器學習用例

ML 在企業和消費者應用程式的使用量越來越高。自 2010 年代中期以來，利用 ML 為消費者提供卓越（或以前不可能的）服務的應用程式呈爆炸式增長。

隨著資訊和服務的爆炸式增長，如果沒有 ML 的幫助，無論是在**搜索引擎**還是**推薦系統**中，我們都很難找到我們想要的東西。當你存取 Amazon 或 Netflix 等網站時，系統會向你推薦最符合你喜好的項目。如果你不喜歡任何推薦項目，你要搜索特定項目，搜索結果可能由 ML 支持的。

如果你有智慧型手機，ML 可能已經在許多日常活動中向你提供幫助。**預測字詞**讓你在手機上打字變得更輕鬆，這是一種機器學習系統，為你建議下一步可能想表達的內容。機器學習系統可能會在你的照片編輯應用程式中運行，以建議如何最好地增強你的照片。你可能會使用指紋或臉部來解鎖手機，這需要 ML 系統來預測指紋或臉部是否與你相匹配。

吸引我進入 ML 領域的用例是**機器翻譯**，這可自動將一種語言翻譯成另一種語言。它能讓來自不同文化的人們相互交流，消除語言障礙。我的父母不會說英語，但有了 Google 翻譯，他們現在可以閱讀我的作品，並跟我那些不會說越南語的朋友們交談。

有了 Alexa 和 Google Assistant 等智慧個人助理，更多 ML 出現在我們家中。智慧安全鏡頭可以在你的寵物離開家裡、或出現不速之客時通知你。一位朋友擔心他年邁的獨居母親（如果她摔倒了，沒有人可以扶她起來）所以他使用了一個家庭健康監測系統，來預測家裡是否有人摔倒。

儘管消費者 ML 應用程式市場正在蓬勃發展，但大多數 ML 用例仍在企業世界內。企業 ML 應用程式往往具有與消費者應用程式截然不同的需求和注意事項。雖然有很多例外，但在大多數情況，企業應用程式可能有更嚴格的準確性要求，但對延遲的要求更寬容。例如，將語音識別系統的準確率從 95% 提高到 95.5% 對大多數消費者來說可能並不明顯，但將資源分配系統的效率提高 0.1% ，就可以幫助像 Google 或 General Motors 這樣的公司節省數百萬美元。同時，一秒鐘的延遲可能會讓消費者分心，並打開了其他東西，但企業用戶對高延遲的容忍度可能更高。對於有興趣用 ML 應用程式創業的人，消費者應用程式更容易發行廣傳、卻難以獲利。而大多數企業用例並不那麼明顯，除非你有相關經歷。

根據 Algorithmia 的 2020 年企業機器學習狀況調查，企業中的 ML 應用程式多種多樣，服務含括內部用例（降低成本、生成客戶洞見和情報、內部處理自動化）與外部用例（改善客戶體驗、留住客戶、與客戶互動），如圖 1-3 所示[8]。

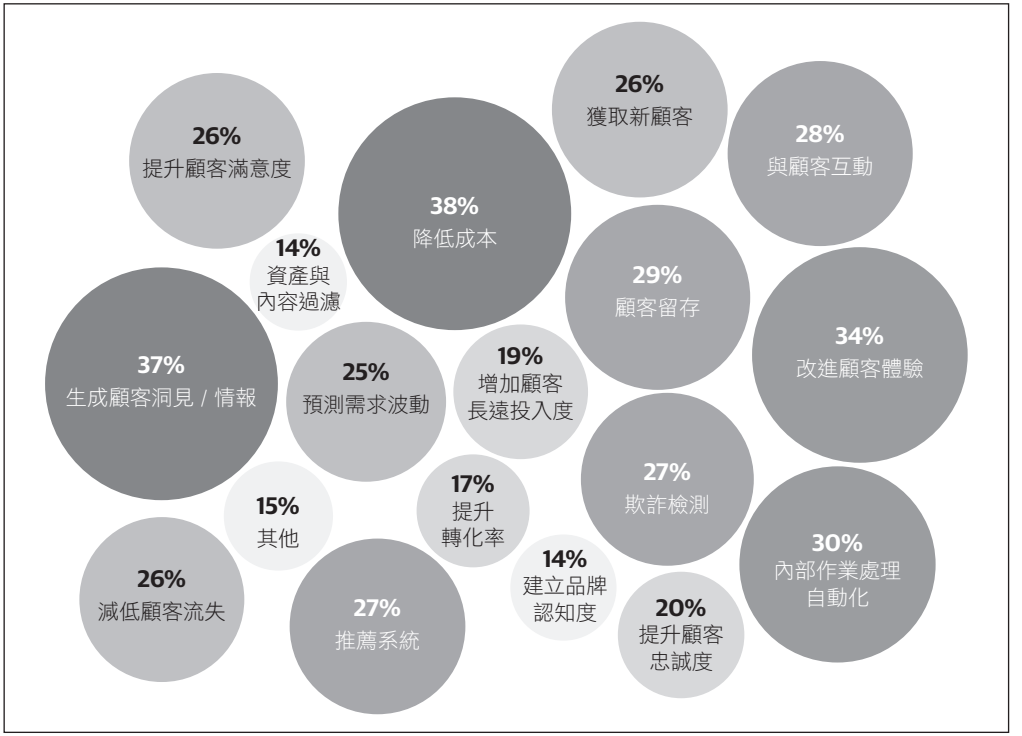

圖 1-3　2020 年企業機器學習狀態。資料來源：改編自 Algorithmia 圖像

詐欺檢測（*fraud detection*）是機器學習在企業界存在最久的應用之一。如果你的產品或服務涉及任何價值的交易，則很容易出現詐欺。透過利用 ML 解決方案進行異常檢測，你的系統可以從歷史詐欺交易中學習，並預測未來交易是否存在詐欺。

8　《2020 State of Enterprise Machine Learning》，*Algorithmia*，2020 年，*https://oreil.ly/wKMZB*。

要決定你的產品或服務收取多少費用，可能是最艱難的商業決策之一，何不讓 ML 幫幫忙？**價格優化**（*price optimization*）是指對於給定目標函數（例如公司的利潤率、收入或增長率），在特定時間段估算價格，以最大化函數的過程。基於 ML 的價格優化最適用於交易量大、需求波動大、消費者願意支付動態價格的情況——例如：互聯網廣告、機票、住宿預訂、拼車、節目活動等。

想要運作好一家企業，預測客戶的需求非常重要，這樣你才能準備預算、庫存、分配資源和更新定價策略。例如，假設你經營一家雜貨店，你希望有足夠的存貨以便顧客找到他們想要的東西，但又不想有過多的存貨，否則過多的存貨可能變質，進而導致虧損。

一名新用戶的獲取成本高昂。截至 2019 年，一款 app 要獲取一位會進行 app 內購買的用戶，平均成本為 86.61 美元[9]。Lyft 的獲取成本估計為 158 美元 / 人[10]。對於企業客戶而言，這一成本要高得多。客戶獲取成本被投資者稱為新創公司殺手[11]。只要少量降低客戶獲取成本，就可大幅增加利潤。這可以透過更好地識別潛在客戶、展示更有針對性的廣告、在合適的時間提供折扣等來實現——這些都是 ML 的合適任務。

耗費大筆資金而獲得的客戶，若是眼睜睜看他們流失就太可惜了。獲取新用戶的成本大約是保留現有用戶的 5 到 25 倍[12]。**流失預測**（*Churn prediction*）是指預測特定客戶何時停止使用你的產品或服務，以便採取適當行動來贏回他們。流失預測不僅可以用於客戶，也可以用於員工。

---

9 《Average Mobile App User Acquisition Costs Worldwide from September 2018 to August 2019, by User Action and Operating System》，*Statista*，2019 年，*https://oreil.ly/2pTCH*。

10 Jeff Henriksen，《Valuing Lyft Requires a Deep Look into Unit Economics》，福布斯，2019 年 5 月 17 日，*https://oreil.ly/VeSt4*。

11 David Skok，《Startup Killer: The Cost of Customer Acquisition》，*For Entrepreneurs*，2018 年，*https://oreil.ly/L3tQ7*。

12 Amy Gallo，《The Value of Keeping the Right Customers》，哈佛商業評論，2014 年 10 月 29 日，*https://oreil.ly/OlNkl*。

防止客戶流失的重點在於，當他們遇到問題時立即解決，讓他們開心。自動化支援個案分類可以協助解決這個問題。以前，當客戶創建支援個案或發送電子郵件時，需要先對其進行處理，然後傳遞給不同的部門，直到個案傳到可以解決它的人手上。ML 系統可以分析個案內容，並預測它應該去哪裡，這可以縮短回應時間，並提高客戶滿意度。它還可用於對內部 IT 個案進行分類。

ML 在企業中的另一個流行用例是品牌監控。品牌是企業的寶貴資產[13]。監測公眾和客戶對品牌的認知非常重要。例如品牌是何時 / 何地 / 如何被提及的，無論是明確的（例如有人提到「Google」時）還是隱含的（例如有人說「搜索巨頭」時），以及與之相關的情緒。如果突然在提及品牌同時伴隨大量負面情緒，則要盡快解決。情感分析是典型的 ML 任務。

最近令人十分振奮的一組 ML 用例發生在醫療保健領域。ML 系統可以檢測皮膚癌和診斷糖尿病。儘管許多醫療保健應用程式面向消費者，但由於程式對準確性和隱私的嚴格要求，其服務通常只透過醫療保健提供者（例如醫院）提供，或只用於協助醫生進行診斷。

了解機器學習系統

了解 ML 系統將有助於其設計和開發。在本節，我們將討論 ML 系統與「研究領域 ML」（或學校經常教授的機器學習）和傳統軟體兩者有何不同，這構成了撰寫本書的動機。

研究與生產環境中的機器學習

由於 ML 在行業中的使用情況還很新，大多數擁有 ML 專業知識的人都是透過學術界獲得的：參加課程、做研究、閱讀學術論文。如果這也是你的背景，那麼你可能很難理解在外部署 ML 系統的挑戰；在大量應對這些挑戰的解決方案中備感吃力的苦苦尋找。生產環境中的 ML 與研究性質的 ML 有很大不同。表 1-1 顯示了五個主要差異。

13 Marty Swant，《The World's 20 Most Valuable Brands》，福布斯，2020 年，*https://oreil.ly/4uS5i*。

表 1-1　研究性質 ML 與生產環境 ML 之間的主要區別

	研究	生產環境
需求	模型在基準資料集的效能達到領先水平	不同利益相關者，不同需求
運算優先級	快速訓練，高吞吐量	快速推理，低延遲
資料	靜態 [a]	常轉移
公平性	通常不是重點	必需考慮
可解釋性	通常不是重點	必需考慮

[a] 一個研究的子領域側重於持續學習：開發模型以處理不斷變化的資料分布。我們將在第 9 章介紹持續學習。

不同的利益相關者和要求

參與研究和「爭取排名」專案的人，通常會有一致的目標。最常見的目標是模型效能——開發一個在基準資料集上實現最領先結果的模型。為了爭取效能上的微小改進，研究人員經常採用使模型過於複雜，失去實用性技術。

ML 系統要進入生產階段，涉及許多利益相關者。每個利益相關者都有自己的需求。當系統有著不同且相互衝突的需求，可能會導致難以設計、開發和選取滿足所有需求的 ML 模型。

假設有一個向用戶推薦餐廳的流動應用程式。該應用程式透過向餐廳收取每筆訂單 10% 的服務費來賺錢。這意味著高價訂單比低價訂單為應用程式帶來的更多收入。該項目涉及 ML 工程師、銷售人員、產品經理、基礎設施工程師和一名經理：

ML 工程師

想要一個用戶最有可能從推薦餐廳結果中下單的模型，相信可以透過使用具有更多資料、更複雜的模型來達標。

銷售團隊

想要一個推薦更高價餐廳的模型，因為這些餐廳會帶來更高的服務費。

產品團隊

注意到每次延遲增加，都會導致服務訂單減少，因此想要一個可以在不到 100 毫秒內返回推薦餐廳的模型。

ML 平台團隊

隨著流量增長，團隊因為現有系統規模化的問題常在半夜醒來，所以想推遲模型更新，以優先改進 ML 平台。

經理

想最大化利潤，實現這一目標的一種方法可能是放棄 ML 團隊[14]。

「推薦用戶最有可能點擊的餐廳」和「推薦能為 app 帶來最多收益的餐廳」是兩個不同的目標，在第 40 頁「解耦目標」一節中，我們將討論如何開發滿足不同目標的 ML 系統。劇透：我們將為每個目標開發一個模型，並結合他們的預測。

現在讓我們想像一下，我們有兩個不同的模型。模型 A 用作推薦用戶最有可能點擊的餐廳，模型 B 用作推薦將為應用程式帶來最多收入的餐廳。A 和 B 可能是非常不同的模型。應該為用戶部署哪種模型？更難做出決定的是，A 和 B 都不滿足產品團隊提出的要求：他們不能在 100 毫秒內返回餐廳推薦。

開發 ML 專案時，ML 工程師必須了解所有利益相關者的需求，以及這些需求的嚴格程度。例如能夠在 100 毫秒內返回推薦是一項必須具備的要求（公司發現，如果模型花費超過 100 毫秒來推薦餐廳，10% 的用戶會失去耐心並關閉應用程式）那麼無論是模型 A 和模型 B 都行不通。但如果這只是一個錦上添花的需求，你可能仍要在模型 A 或模型 B 之間做出選擇。

成功的研究專案不總是用於生產環境，原因之一是生產與研究有著不同需求。例如，集成（ensembling）是一種在許多 ML 競賽（包括著名的 100 萬美元 Netflix 獎）獲勝者常用的技術，但它並未在生產環境廣泛應用。集成結合了「多種學習算法，以獲得比單獨算法組件更好的預測效能」[15]。雖然集成可以為你的 ML 系統帶來小幅度效能提升，但此法往往使系統過於複雜而無法用於生產環境，例如較慢的預測速度，或結果較難解釋。我們將在第 158 頁「集成」小節進一步討論。

---

14 機器學習和資料科學團隊在公司大規模裁員期間率先離職並不罕見，例如：IBM（*https://oreil.ly/AfUB5*）、Uber（*https://oreil.ly /t0QpY*）、Airbnb（*https://oreil.ly/q4M4E*）。另請參閱 Sejuti Das 的分析《How Data Scientists Are Also Susceptible to the Layoffs Amid Crisis》，*Analytics India Magazine*，2020 年 5 月 21 日，*https://oreil.ly/jobmz*。

15 維基百科，s.v.《Ensemble learning》，*https://oreil.ly/5qkgp*。

對於許多任務，效能的小幅度改進可能會大大提高收入或節約成本。例如，產品推薦系統的點擊率提高 0.2%，可能會使電子商務網站的收入增加數百萬美元。但對於許多任務，用戶可能注意不到這些小改進。如果一個簡單模型可以完成合理的工作，那麼複雜模型帶來的效能提升必須有顯著性，才能合理化其複雜性。

對 ML 排行榜的批評

近年來，有很多人對 ML 排行榜提出批評，包括 Kaggle 等競賽，還有 ImageNet 或 GLUE 等研究排行榜。

一個顯而易見的論點是，這些競賽已經為你完成了許多構建 ML 系統所需的困難步驟。[16]。

另一個不太明顯的論點是，當你有多個團隊在同一個保留測試集上進行測試時，會發生多重假設測試場景，因此這可能只是碰巧有個模型，比其他模型做得更好 [17]。

研究人員已經注意到研究與生產之間的利益錯位。Ethayarajh 和 Jurafsky 在 EMNLP 2020 論文中認為，效能基準推動自然語言處理（NLP）進步，是在忽視了從業者看重其他品質（例如精巧程度、公平性和能源效益）[18] 的情況下，激勵人們創建「更準確」的模型。

計算優先級

設計 ML 系統時，過於重視模型開發，對模型部署和維護部分的關注不足，是沒有部署過 ML 系統的人常犯錯誤之一。

16 Julia Evans，《Machine Learning Isn't Kaggle Competitions》，2014 年，*https://oreil.ly/p8mZq*。

17 Lauren Oakden-Rayner，《AI Competitions Don't Produce Useful Models》，2019 年 9 月 19 日，*https://oreil.ly/X6RIT*。

18 Kawin Ethayarajh 和 Dan Jurafsky，《Utility Is in the Eye of the User: A Critique of NLP Leaderboards》，EMNLP , 2020, *https://oreil.ly/4Ud8P*。