前言

近年來，生命科學和資料科學已經融合。機器人和自動化的進步使化學家和生物學家能夠生成大量數據。今天的科學家能夠在一天內產生比 20 年前整個職業生涯更多的數據。這種快速生成數據的能力也帶來了許多新的科學挑戰。我們不再處於將數據載入到電子表格中並製作幾個圖表來處理數據的時代。為了從這些數據集中提取科學知識，我們必須能夠識別和擷取非顯而易見的關係。

過去幾年出現的**深度學習**是識別數據模式和關係的強大工具，這是徹底改變圖像分析、語言翻譯、語音識別等問題的演算法。深度學習演算法擅長識別和利用大型數據集的模式。由於這些原因，深度學習在生命科學領域中有廣泛的應用。本書說明深度學習如何應用於遺傳學、藥物開發、醫學診斷等多個領域。我們說明的案例都附有程式碼範例，這些範例提供了對方法的實際介紹，並為讀者提供了未來研究和探索的起點。

本書編排慣例

下列為本書的編排慣例：

斜體字（*Italic*）

用來表示新術語、URL、電郵信箱、檔案名稱與附加檔名。中文以楷體表示。

定寬字（`Constant width`）

用來表示樣式碼或在段落中表示如變數或函式名稱、資料庫、資料型別、環境變數、敘述與關鍵字等程式元素。

第一章

為何是生命科學？

雖然數據有許多技術方向可以研究，但很少有領域有著像生物醫學研究一樣的影響。現代醫學的出現從根本上改變了人類生存的本質。在過去的 20 年中，我們看到了改變無數個人生活的創新。HIV/AIDS 於 1981 年首次出現時是一種致命的疾病，持續開發抗逆轉錄病毒療法大大延長了已開發國家患者的預期壽命。現在可以治癒的 C 型肝炎等的疾病，在十年前被認為基本上是無法治癒的。遺傳學的進步能夠識別並很快治療各種疾病，診斷和儀器方面的創新使醫生能夠針對識別和標定人體疾病。許多突破已經從計算方法中受益並繼續發展。

為何深度學習？

機器學習演算法是線上購物到社交媒體等各個方面的關鍵組成部分，計算機科學家團隊正在開發演算法使數位助理（如 Amazon Echo 或 Google Home）能夠理解語音。機器學習的進步使網頁與語音能夠進行即時翻譯。除了機器學習對日常生活的影響外，它還影響了物理和生命科學的許多領域。演算法應用在探測望遠鏡圖像中的新星系與大型對撞機的亞原子相互作用分類等各種領域。

推動這些機器學習技術進步的力量之一是深度神經網路。雖然人工神經網路的技術在 1950 年代發明並於 1980 年代取得進步，直到過去 10 年間電腦硬體的進步才真正實現了此技術的能力。下一章會更深入討論深度神經網路，但先要認識一些深度學習應用的發展：

- 語音識別的許多發展已經在手機、電腦、電視、其他網路設備中普遍存在，這些都是深度學習的結果。
- 圖像識別是自動駕駛汽車、網路搜索、其他應用程序的關鍵組成部分。推動消費者應用的深度學習發展現在被用於生物醫學研究，例如腫瘤細胞分類。
- 推薦系統已成為線上體驗的關鍵組成部分。像 Amazon 這樣的公司使用深度學習以“購買此產品的客戶也買了某某產品”的方法來推動額外消費。Netflix 使用類似的方法來推薦個人可能想要觀看的電影。這些推薦系統背後的許多想法被用於識別可能為藥物開發工作提供起點的新分子。
- 語言翻譯曾經是非常複雜的基於規則的系統。在過去幾年中，深度學習驅動的系統的表現優於經過多年手動調整的系統。許多相同的想法現在被用於從科學文獻中擷取概念並提醒科學家們他們可能錯過的期刊文章。

這些只是應用深度學習方法產生的一些創新。我們收集廣泛可用的科學數據和處理數據的方法時，我們正處在一個有趣的時刻。能夠將數據與新方法結合起來學習數據模式的人可以取得重大的科學進步。

現代生命科學是數據

如前述，生命科學的基本性質已發生變化。機器人技術和小型化實驗的可用性讓生成的實驗數據量顯著增加。在 1980 年代，生物學家進行單一實驗並產生單一結果，這種資料通常可以在計算器的幫助下手動計算。今天的生物學儀器能夠在一兩天內產生數百萬個實驗數據，基因測序等可以產生巨大數據集的實驗已經變得廉價且普遍。

基因測序的進步導致了資料庫的建立，這些資料庫將個體的遺傳密碼與多種和健康相關的結果聯繫起來，包括糖尿病、癌症、囊性纖維化等遺傳性疾病。透過使用計算技術分析和挖掘這些數據，科學家們正在開發對這些疾病的原因的理解，並利用這種理解來開發新的治療方法。

曾經主要依賴於人類觀察的學科現在正使用無法手動分析的數據集。現在，機器學習經常用於對細胞圖像進行分類。這些機器學習模型的輸出用於識別癌症腫瘤，並對其進行分類與評估潛在疾病治療的效果。

實驗技術的進步導致了若干資料庫的發展，這些資料庫對化學品的結構及這些化學品對廣泛的生物過程或活動的影響進行了分類。這些結構 - 活動關係（SAR）構成了稱為化學資訊學或**化學信息學**的基礎。科學家們挖掘了這些大型數據集，並利用這些數據建立預測模型以推動下一代藥物開發。

隨著這些大量數據的出現，需要一種在科學和計算領域都很行的新型科學家。具有這些混合技能的人有可能解開大型數據集的結構和趨勢，並做出明天的科學發現。

你會學到什麼？

本書的前幾章討論深度學習的概述以及它如何應用於生命科學。我們從機器學習開始，機器學習被定義為“從數據中學習的程式設計科學（和藝術）”[1]。

第 2 章簡短介紹深度學習。我們首先舉例說明如何使用這種類型的機器學習來執行線性回歸等簡單任務，然後進入通常用於解決生命科學中的現實問題的更複雜模型。機器學習的進行通常將數據集拆開成用於生成模型的訓練集和用於評估模型性能的測試集，第 2 章會討論有關預測模型的訓練和驗證的一些細節。生成模型後，通常可以透過改變稱為**超參數**的許多特徵讓表現最佳化，這一章概述此過程。深度學習不是單一技術，而是一套相關的方法。第 2 章最後介紹一些最重要的深度學習變化版本。

第 3 章介紹 DeepChem，這是一個開源函式庫，專門用於簡化為各種生命科學應用建立的深度學習模型。介紹 DeepChem 後，我們以第一個程式範例展示如何使用 DeepChem 函式庫生成預測分子毒性的模型。第二個程式範例展示 DeepChem 如何用於對圖像進行分類，這是現代生物學中的一項常見任務。如前述，深度學習用於各種成像應用，從癌症診斷到青光眼檢測，對特定應用的討論激發了一些深度學習方法內部機制的解釋。

第 4 章討論如何將機器學習應用於分子。我們首先介紹分子，這是我們周圍一切的基石。雖然分子可以被認為類似於積木，但它們並不是剛性的。分子柔韌並表現出動態行為。為了使用像深度學習這樣的計算方法來表示分子，我們需要找到一種在計算機中表示分子的方法。這些編碼類似於以一組像素表示圖像的方式。第 4 章的後半部分描述表示分子的多種方式，以及這些表示如何用於建構深度學習模型。

---

1 Furbush, James.“Machine Learning: A Quick and Simple Definition.” *https://www.oreilly.com/ideas/machine-learning-a-quick-and-simple-definition.* 2018.

第 5 章介紹生物物理學領域，它將物理定律應用於生物現象。我們首先討論讓生命成為可能的蛋白質分子機器。預測藥物對身體影響的一個關鍵因素是了解它們與蛋白質的相互作用。為了理解這些影響，我們首先討論蛋白質的建構方式以及蛋白質結構的差異。蛋白質的 3D 結構決定其生物學功能的實體。對於機器學習模型來預測藥物分子對蛋白質功能的影響，我們需要以可由機器學習程序處理的形式表示 3D 結構。第 5 章的後半部分探討表現蛋白質結構的多種方式。有了這些知識，我們再檢視另一個程式範例，以深度學習來預測藥物分子與蛋白質相互作用的程度。

遺傳學已成為當代醫學的重要組成部分。腫瘤的基因測序已經實現了癌症的個性化治療並有可能徹底改變醫學。基因測序曾經是一個需要巨額投資的複雜過程，現在已經司空見慣，而且可以定期進行。我們甚至達到了狗主人可以進行基因測試以確定他們的寵物血統的程度。第 6 章討論遺傳學和基因組學，首先介紹 DNA 和 RNA，這些模板用於生成蛋白質。最近的發現顯示 DNA 和 RNA 的相互作用比最初認為的要複雜得多。第 6 章的後半部有幾個程式範例展示如何使用深度學習來預測影響 DNA 和 RNA 相互作用的許多因素。

這一章的前面部分提到了透過將深度學習應用於生物和醫學圖像分析而取得的許多進展。在這些實驗中研究的許多現象太小而不能被人眼觀察到。為了獲得深度學習方法所使用的圖像，我們需要使用顯微鏡。第 7 章討論無數形式的顯微鏡，從我們在學校使用的簡單光學顯微鏡到能夠以原子解析度獲得圖像的複雜儀器。這一章還討論當前方法的一些局限性，並提供了用於獲取驅動深度學習模型圖像的實驗管道資訊。

深度學習應用於醫學診斷是很有前途的領域。醫學非常複雜，沒有醫生可以親自體現所有可用的醫學知識。在理想情況下，機器學習模型可以消化醫學文獻並幫助醫療專業人員進行診斷。雖然我們還沒有達到這一點，但已經採取了一些積極措施。第 8 章從醫學診斷的機器學習方法的歷史開始，並說明從手工編寫規則到醫學結果統計分析的過渡。與我們討論的許多主題一樣，關鍵部分是以機器學習程序能處理的格式表示醫療資訊。這一章會介紹電子健康記錄以及圍繞這些記錄的一些問題。在許多情況下，醫學圖像可能非常複雜，即使是熟練的人類專家也難以對這些圖像進行分析和解釋。在這些情況下，深度學習可以透過分類圖像和識別關鍵特徵來增強人類分析師的技能。第 8 章總結了一些深度學習如何用於分析來自各個領域的醫學圖像的例子。

如前述，機器學習正在成為藥物開發工作的關鍵組成部分。科學家使用深度學習模型來評估藥物分子和蛋白質之間的相互作用。這些相互作用可以引發對患者具有治療效果的生物反應。我們到目前為止討論過的模型是**判別模型**。輸入一組分子的特徵，該模型能產生一些屬性的預測。這些預測需要輸入的分子可來自大型分子資料庫或科學家的想像。若不是依靠現有的東西而是有一個可以“發明”新分子的計算機程序呢？第 9 章討論一種稱為**生成模型**的深度學習程序。生成模型最初在一組現有分子上訓練，然後用於產生新分子。產生這些分子的深度學習程序也可能受到預測新分子活性的其他模型的影響。

到目前為止我們以“黑盒子”討論深度學習模型，我們使用一組輸入數據呈現模型生成預測而沒有解釋預測的生成方式或原因。在許多情況下，這種類型的預測可能不是最佳的。如果有一個醫學診斷深度學習模型，我們必須了解診斷背後的推理。解釋診斷的推理能使醫生對預測更有信心，也可能影響治療決策。深度學習的一個歷史缺點是模型雖然經常可靠但很難解釋，目前正在開發許多技術以使用戶能夠更好地理解產生預測的因素。第 10 章討論了一些用於使人們理解模型預測的技術。預測模型的另一個重要方面是模型預測的準確性，了解模型的準確性可以幫助我們確定依賴該模型的程度。鑑於機器學習有潛力可進行挽救生命的診斷，因此理解模型準確性至關重要。第 10 章的最後一節討論可用於評估模型預測準確性的一些技術。

第 11 章使用 DeepChem 提供了一個真實的案例研究。在這個例子中，我們使用一種稱為虛擬篩選的技術來識別發現新藥的潛在起點。藥物發現是一個複雜的過程，通常以稱為**篩選**的技術開始。篩選用於鑑定可以最佳化以最終產生藥物的分子。篩選可以透過實驗進行，其中數百萬個分子在稱為檢測（assay）的小型化生物測試中進行測試，或者在使用虛擬篩選的計算機中進行測試。在虛擬篩選中，使用一組已知藥物或其他生物活性分子來訓練機器學習模型，然後使用該機器學習模型來預測大量分子的活性。由於機器學習方法的快速，通常可以在幾天的計算機時間內處理數百萬分子。

本書的最後一章探討深度學習在生命科學中的影響和未來潛力。討論當前工作的一些挑戰，包括數據集的可用性和品質。我們還強調了其他一些領域的機會和潛在缺陷，包括診斷、個人化醫療、藥物開發、生物學研究。

第二章

深度學習介紹

這一章的目標是介紹深度學習的基本原理，熟悉深度學習的讀者可略過。若經驗不多則應該仔細研讀這一章，它涵蓋閱讀本書其餘內容的基本知識。

我們討論的大部分題目需要建立一個數學函數：

$$\mathbf{y} = f(\mathbf{x})$$

注意 **x** 與 **y** 是粗體字。這表示它們是向量。函數的輸出入數字可多達數百萬。下面是一些函數的例子：

- **x** 是圖形中每個像素的顏色。$f(\mathbf{x})$ 等於 1 表示圖形是貓，否則為 0。
- 同上，但 $f(\mathbf{x})$ 應該是數字向量。第一個元素表示圖形是否為貓、第二個表示圖形是否為狗、第三個表示是否為飛機，依此類推幾千種不同的物件。
- **x** 是基因的 DNA 序列。**y** 是與基因中鹼基數量相同的向量。如果該鹼基是編碼蛋白質區域的一部分，則每個元素應等於 1，否則應為 0。
- **x** 描述一個分子的結構（後面章節會討論各種分子表示方法）。**y** 是描述分子物理屬性的元素的向量：如是否容易水解、鍵強度等。

如你所見，$f(\mathbf{x})$ 可以是一個非常複雜的函數！它通常需要一個長向量作為輸入並嘗試從中擷取不明顯的資訊。

解決這個問題的傳統方法是手工設計一個功能。從分析問題開始，什麼樣的像素模式表明貓的存在？什麼樣的 DNA 模式區分編碼區和非編碼區？你可以編寫電腦程式以識別特定類型的特徵，然後嘗試識別可產生所需結果的特徵組合。這個過程緩慢並耗費人力，且在很大程度上取決於執行者的專業知識。

機器學習採用完全不同的方法。你可以讓計算機根據數據學習出函數而不是手動設計函數。你收集千百萬個圖像，讓每個圖像都標記是否有貓。你將所有訓練數據輸入計算機並讓它搜索一個函數，該函數對於有貓的圖像始終接近 1，對於沒有貓的圖像則接近 0。

“讓計算機搜索出函數”是什麼意思？通常就是建立一個定義一些大分類函數的**模型**。此模型包括**參數**，可以採用任何值的變數。透過選擇參數的值，可以從模型定義的類別中的所有函數中選擇一個特定函數。計算機的工作是選擇參數的值，它試圖找到訓練數據用作輸入時，輸出盡可能接近相應的目標的值。

線性模型

最簡單的模型是線性模型：

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{b}$$

等式中的 **M** 是矩陣（有時稱為“權重（weight）”）而 **b** 是向量（稱為“偏差（bias）”）。它們的大小由輸出入值的數量決定。若 **x** 的大小為 T 且你想要長度 S 的 **y**，則 **M** 是 S × T 矩陣，且 **b** 是長度為 S 的向量。它們一起組成此模型的參數，這個等式只是表示每個輸出元素是輸入元素的線性組合。你透過設定參數（**M** 與 **b**）選擇每個元素的線性組合。

這是最早的機器學習模型之一。它於 1957 年被引入，稱為**感知器**（*perceptron*）。這個名字是一個驚人的行銷活動：聽起來很科幻，似乎是個美妙的事物，而事實上它只不過是一個線性變換。無論如何，這個名字已經成功地堅持了半個多世紀。

線性模型很容易以完全通用的方式表達。無論應用在什麼問題上，它都具有完全相同的形式。線性模型之間的唯一差異是輸入和輸出向量的長度。此後只需選擇參數值，這可以透過通用演算法以簡單的方式完成。這正是我們對機器學習的期望：模型和演算法獨立於嘗試解決的問題外。只需提供訓練數據就會自動確定參數，將通用模型變換為解決問題的函數。

不幸的是線性模型限制很大。如圖 2-1 所示，線性模型（在一個維度上，也就是一條直線）根本無法符合大多數真實數據集。轉移到非常高維的數據時問題變得更加嚴重。圖像中的像素值的線性組合不能可靠的識別圖像是否包含貓。這種任務需要更複雜的非線性模型。實際上，任何解決這種問題的模型都必然非常複雜且非常非線性。但是我們如何以通用的方式編寫它呢？所有可能的非線性函數的空間都是無限複雜的。我們如何定義一個模型，只需選擇參數值就可以建立任何我們想要的非線性函數？

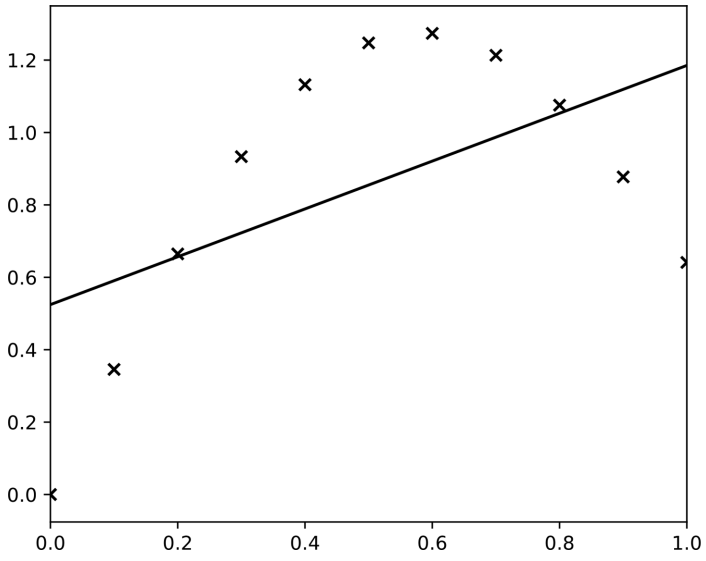

圖 2-1　一個線性模型無法符合曲線上的數據點，這需要非線性模型。

多層感知器

一種節點的做法是前後堆疊多個線性變換。例如：

$$\mathbf{y} = \mathbf{M}_2\varphi(\mathbf{M}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

仔細觀察我們做了什麼。從一個普通線性變換 $\mathbf{M}_1\mathbf{x} + \mathbf{b}_1$ 開始，然後將結果傳給非線性的 $\varphi(x)$，再傳給第二個線性變換。稱為**激勵**（*activation*）**函數**的 $\varphi(x)$ 是運作基礎，沒有它則模型還是線性的，不會比前面的函數更好。線性組合的線性組合不會比原始輸入的線性組合好！加入非線性讓模型可以學習出更廣泛的函數。

不限於兩層線性變換，我們可以依需要堆疊任意數量的變換：

$$\mathbf{h}_1 = \varphi_1(\mathbf{M}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{h}_2 = \varphi_2(\mathbf{M}_2\mathbf{h}_1 + \mathbf{b}_2)$$

$$\cdots$$

$$\mathbf{h}_{n-1} = \varphi_{n-1}(\mathbf{M}_{n-1}\mathbf{h}_{n-2} + \mathbf{b}_{n-1})$$

$$\mathbf{y} = \varphi_n(\mathbf{M}_n\mathbf{h}_{n-1} + \mathbf{b}_n)$$

這種模型稱為**多層感知器**或 MLP。中間的步驟 h_i 稱為**隱藏層**，因為它們並非輸入或輸出，只是計算結果的程序中的中間值。還要注意到每個 $\varphi(x)$ 的下標，它表示不同層可能需要不同的非線性。

你可以將此計算視為一堆圖層，如圖 2-2 所示。每層對應一個線性變換跟著一個非線性。資訊從一層流到另一層，前一層的輸出作為下一層的輸入。每一層都有自己的一組參數用於決定如何根據輸入計算輸出。

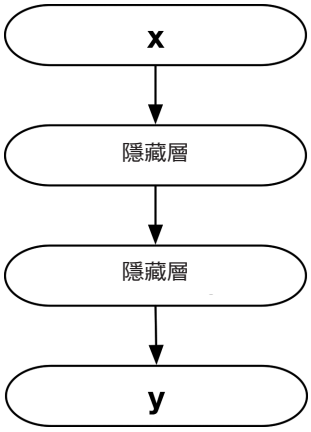

圖 2-2　多層感知器，資訊從一層流動到另一層。

多層感知器與變數又稱為**神經網路**，此名稱反映機器學習與神經生物學之間的相似之處。生物神經元連接到其他神經元，它接收來自它們的信號、將信號相加、然後根據結果發出自己的信號。你可以將 MLP 的工作方式粗略視為與大腦中的神經元一樣！

激勵函式 $\varphi(\mathbf{x})$ 應該是什麼？答案居然是不重要。當然，並不是完全不重要。雖然看起來不重要，但不像你預期的那樣。幾乎任何合理的函數（單調、平滑）都行。多年來已經嘗試了許多不同的函數，雖然有些函數比其他函數更好，但幾乎所有函數都可以產生不錯的效果。

當今最流行的激勵函式可能是**線性整流函數**（ReLU），$\varphi(x) = \max(0, x)$。如果你不確定使用什麼函式，這可能是一個很好的預設值。其他常見的選擇包括**雙曲正切** $\tanh(x)$ 和 *S* **函數**，$\varphi(x) = 1/(1 + e^{-x})$。這些函數如圖 2-3 所示。

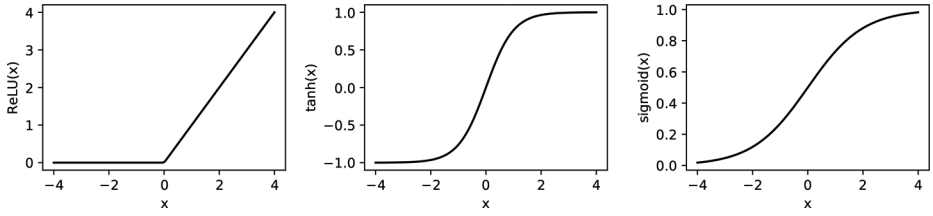

圖 2-3　三種常見的激勵函式：線性整流、雙曲正切、sigmoid。

我們還必須為 MLP 選擇另外兩個屬性：**寬**和**深**。簡單的線性模型沒有選擇，**x** 和 **y** 的長度決定 **M** 和 **b** 的大小。隱藏層則不是這樣，寬是指隱藏層的大小，我們可以任意選擇每個 $\mathbf{h}_i$ 的長度。視問題而定，你可能希望它們比輸入和輸出向量大或小。

深指模型層數量。只有一個隱藏層的模型稱為**淺**（*shallow*）。有多個隱藏層的模型稱為**深**（*deep*），這就是“深度學習”一詞的由來；它表示“使用多層模型的機器學習”。

選擇模型層的數量和寬度涉及的技巧與科學一樣多。或者更正式的說法是：“這還是一個有待研究的領域”。通常它只是嘗試許多組合，看看哪些有效。然而，有一些原則可以提供指導，或者至少可以幫助你了解事後的結果：

1. 具有一個隱藏層的 MLP 是**通用近似**。

   這意味著它可以完全逼近任何函數（在某些合理的限制範圍內）。從某種意義上說，你永遠不需要多個隱藏層，這已足以重現任何函數。不幸的是這個結果帶來了一個重要的警告：近似的精確度取決於隱藏層的寬，你可能需要一個非常寬的層以獲得足夠的精度。這帶出第二個原則。

2. 深模型往往比淺模型需要更少的參數。這個陳述刻意含糊不清。

   對於特定的特殊情況，可以證明更嚴格的陳述，但它仍然適用於一般準則。可能更好的說明方式是：每個問題都需要一個具有一定深度的模型才能有效的達到可接受的精度。在較淺的深度處，所需的層寬（以及參數的總數）迅速增加。這聽起來像你應該用深模型而不是淺模型。不幸的是，它與第三項原則部分矛盾。

3. 深模型往往比淺模型更難訓練。

   直到 2007 年左右，大多數機器學習模型都很淺。深模型的理論優勢是眾所周知的，但研究人員通常無法對其進行訓練。從那時起，一系列進步逐漸提高了深模型的實用性。這包括更好的訓練演算法、更容易訓練的新型模型、當然還有更快的計算機與更大的數據集相結合，這些進步引發“深度學習”領域。然而，儘管有所改進，但一般原則仍然是正確的：深的模型往往比淺模型更難訓練。

訓練模型

這帶出下一個主題：如何訓練模型？ MLP 為我們提供了一個（大多數）通用模型，可用於任何問題（稍後會討論其他更特殊的模型）。現在我們想要一個類似的通用演算法來找到特定問題的模型參數的最優值。我們怎麼做？

當然，首先需要一系列數據來訓練它，該數據集稱為**訓練集**，它應該由稱為**樣本**的大量 **(x,y)** 對組成。每個樣本都指定模型的輸入，以及在該輸入下你希望模型的輸出。舉例來說，訓練集可以是圖像的集合加上表示每個圖像是否有貓的標籤。

接下來需要損失（loss）函式 $L(\mathbf{y}, \hat{\mathbf{y}})$，$\mathbf{y}$ 是模型的實際輸出，而 $\hat{\mathbf{y}}$ 是訓練集設定的目標值。這是評估模型是否很好的複製訓練數據的方法。然後它與訓練集的每個樣本平均：

$$\text{平均損失} = \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$

$L(\mathbf{y}, \hat{\mathbf{y}})$ 應該在參數接近時小而遠離時大。換句話說，我們試著用訓練集的每個樣本作為模型輸入，並檢視輸出與目標值的距離後與整個訓練集平均。

每個問題都需要選擇合適的損失函式。常見的選擇是歐氏距離（Euclidean distance，又稱為 L_2 距離），$L(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\Sigma_i (y_i - \hat{y}_i)^2}$（此運算式中的 y_i 表示 $\mathbf{y}$ 向量的第 i 個元素）。以 $\mathbf{y}$ 代表機率分佈時最常見的選擇是交叉熵（cross entropy），$L(\mathbf{y}, \hat{\mathbf{y}}) = -\Sigma_i y_i \log \hat{y}_i$。也有其他選項且沒有通用的“最佳”選項，視問題的細節而定。

有了評估模型表現的方法後，我們需要改善它的方法。我們想要找尋讓訓練集平均損失最小的參數值，方法有很多種，但大部分深度學習使用某種**梯度下降**（*gradient descent*）演算法。以 θ 表示模型的所有參數集合。梯度下降採取一系列小步驟：

$$\theta \leftarrow \theta - \epsilon \frac{\partial}{\partial \theta} \langle L \rangle$$

$\langle L \rangle$ 是訓練集平均損失。每個步驟“向下”移動一點距離。模型的參數一點點改變，目標是讓平均損失下降。若天時地利人和，則最終會產生能解決問題的參數。ϵ 稱為**學習率**，它決定每一步改變參數的大小。它需要很小心的選擇：值太小導致學習非常慢，太大則會妨礙演算法學習。

這個演算法確實有效，但有一個嚴重的問題。對於梯度下降的每一步，我們都需要遍歷訓練集的每個樣本。這意味著訓練模型所需的時間與訓練集的大小成正比！假設訓練集有一百萬個樣本，計算一個樣本的損失梯度需要一百萬個操作，並且需要一百萬步才能找到一個好的模型（這些數字都是真正的深度學習應用程序的典型代表）。因此訓練需要**萬兆**（*quintillion*）個運算。這需要相當長的時間，即使在快速的計算機上也是如此。

幸好有更好的方法：以較少的樣本做平均來評估 $\langle L \rangle$。這是**隨機**（*stochastic*）**梯度下降**（SGD）演算法的基礎。每個步驟從訓練集取一小量樣本（稱為**批次**，*batch*）並計算損失函式的梯度，只對批次中的樣本做平均。我們可以將此視為在整個訓練集上取平均值時所得到的估計，儘管可能是噪音非常多的估計。我們執行一個梯度下降步驟，然後為下一步選擇一批新樣本。

這種演算法會比較快。每個步驟所需的時間只看批次的大小，而批次可以很小（通常是數百個樣本）且與訓練集的大小無關。缺點是每個步驟降低的損失較少，因為它是根據噪音而非真正的梯度來評估梯度。然而整體訓練時間還是比較少。

大多數的深度學習使用的最佳化演算法都基於 SGD，但有許多變體以不同的方式對其進行改進。幸好你通常可以將這些演算法視為黑盒子，並相信它們可以做正確的事情而無需了解它們運作的所有細節。目前兩種最流行的演算法稱為 Adam 和 RMSProp。如果對使用什麼演算法有疑問，那麼其中任何一個演算法都可能是合理的選擇。

驗證

假設你已完成前述內容，收集了大量的訓練數據、選擇一個模型、然後執行訓練演算法直到損失變得非常小。恭喜，您現在有一個函式可以解決你的問題！

真的嗎？

對不起，並非如此簡單！你真正能確定的是該函式對**訓練數據**上執行良好。你希望它也可以很好的處理其他數據，但還不能確定。接下來必須驗證模型以確定它是否適用於沒訓練過的數據。

因此你需要第二個數據集，稱為**測試集**。它具有與訓練集完全相同的形式，即 $(\mathbf{x}, \mathbf{y})$ 對的集合，但兩者應該沒有相同的樣本。你以訓練集訓練模型，然後在測試集上進行測試。這帶給我們機器學習最重要的原則之一：

- 設計或訓練模型時絕對不能用到測試集。

實際上，測試集的數據最好連看都不看。測試集數據僅用於測試經過完全訓練的模型以了解其表現狀況。如果讓測試集以任何方式影響模型，則可能會產生在測試集上的表現比從未遇過的數據更好的模型。它不再是一個真正的測試集，而是另一種訓練集。

這與稱為**過適**（*overfitting*）的數學概念有關。訓練數據應該代表更大的數據分佈，即可能想要使用該模型的所有輸入的集合。但是你不能用所有可能的輸入訓練它。你只能建立一組有限的訓練樣本，在這些樣本上訓練模型並希望它能夠學習在其他樣本上同樣有效的一般策略。過適是指訓練時過度適應訓練樣本特有的特徵，使得模型在訓練樣本上較其他樣本的表現更好。

正規化

過適對機器學習是一個主要問題。因此已經開發了許多技術來避免它。這些技術統稱為**正規化**（*regularization*）。任何正規化技術的目標都是避免過適，並產生一個適用於任何輸入而不僅僅是用於訓練的特定輸入的模型。

在我們討論特定的正規化技術之前，有兩個理解重點。

首先，避免過適的最佳方法**幾乎總是**取得更多的訓練數據。訓練集越大，表示“真實”數據分佈越好，學習演算法過適的可能性就越小。當然有時是不可能的：也許你根本無法獲得更多數據，或者收集數據可能非常昂貴。在這種情況下，你只需要盡可能地利用你擁有的數據，如果過適是一個問題，您將不得不使用正規化來避免它。但是更多的數據可能會比正規化有更好的結果。

其次，沒有通用的“最佳”方式進行正規化。這一切都取決於問題。畢竟，訓練演算法並不知道它過適，它所知道的只是訓練數據。它不知道真實的數據分佈與訓練數據有什麼不同，因此它只能產生一個在訓練集上表現良好的模型。如果那不是你想要的，那就由你來告訴它。

這是任何正規化方法的本質：讓訓練過程偏差以使某些類型的模型優於其他模型。你預設“好”模型應具有哪些屬性以及它與過適模型的區別，然後告訴訓練演算法優先選擇具有這些屬性的模型。當然，這些預設通常是隱含的而不是明確的。透過選擇特定的正規化方法，預設可能並不明顯，但總是在那裡。

最簡單的正規化方法之一就是用較少的步驟訓練模型。在訓練初期，它往往會接受可能適用於真實分佈的訓練數據的粗略屬性。執行的時間越長就越有可能開始了解訓練樣本的細節。限制訓練步數可以降低過適的機會，更正式的說，你預設的“好”參數值不應該與訓練的任何值有太大的不同。

另一種方法是限制模型參數的大小。舉例來說，你可以在損失函數中添加一個與 $|\theta|^2$ 成比例的項，其中 θ 是包含所有模型參數的向量。如此則預設“好”參數值不應大於必要值，它反映了過適（雖然不一定）涉及一些參數變得非常大的事實。

一種非常流行的正規化方法稱為**丟棄**（*dropout*）。它涉及做一些起初看起來很荒謬但實際上效果出奇的好的事情。對模型中的每個隱藏層隨機選擇輸出向量 h_i 中的元素子集，並將它們設為 0，在梯度下降的每個步驟中選擇不同的隨機元素子集。這看起來只是破壞模型：內部計算隨機設為 0 時如何指望它可行？丟棄原理的數學理論有點複雜，簡單說就是假設模型中的個別計算不應該太重要。你應該能夠隨機刪除任何個別計算且模型的其餘部分應該在沒有它的情況下繼續運作，這迫使它學習冗餘、高度分散的數據表示，使得過適變得不可能。如果不確定要使用哪種正規化方法，則先嘗試使用丟棄。

超參數最佳化

前面看到有很多選擇要做，即使所謂使用“通用”學習演算法的通用模型也是如此。例如：

- 模型中的層數
- 每個層的寬
- 要執行的訓練步驟數量
- 訓練期間使用的學習率
- 丟棄時要設為 0 的元素的比例

這些選項稱為**超參數**（*hyperparameter*）。超參數是模型或訓練演算法中必須事先設定，而不是由訓練算法學習設定。但要如何選擇——根據數據自動選擇不就是機器學習的目的嗎？

這帶出**超參數最佳化**的主題。最簡單的方法就是為每個超參數嘗試很多值，看看哪種方法效果最好。想要為多個超參數嘗試大量值時，成本會變得非常昂貴，因此有更複雜的方法，但基本思想保持不變：嘗試不同的組合，看看什麼效果最好。

但是要如何分辨好壞？最簡單的答案是只看在訓練集上產生損失函數（或其他一些準確度）的最低值，但要記住這不是我們真正關心的。我們希望最小化測試集而不是訓練集的誤差，這對影響正規化的丟棄率等超參數尤其重要。較低的訓練集誤差可能僅意味著模型過適，對訓練數據最佳化。因此，我們想要嘗試大量的超參數值，然後使用讓測試集損失最小的值。

但我們不能這樣做！要記住：設計或訓練模型時不得以任何方式使用測試集。它的工作是告訴你模型可能對它以前從未見過的新數據有效。僅僅因為一組超參數在測試集上發揮最佳表現，並不能保證這些值始終能夠表現的最好。我們不能讓測試集影響模型，否則它不再是無偏差的測試集。

解決方法是建立另一個數據集，稱為**驗證集**。它不得與訓練集或測試集有任何相同的樣本。完整程序如下：

1. 以每組超參數值在訓練集上訓練模型，然後計算驗證集上的損失。
2. 無論哪一組超參數都能在驗證集上得到最低損失時，以它們作為最終模型。
3. 以測試集在最終模型執行以獲得其表現最公正的評估。

其他模型類型

這仍然需要另外做出一個決定，它本身就是一個巨大的主題：使用什麼類型的模型。前面介紹過多層感知器，它們的優點是可以應用於許多不同問題的通用模型。不幸的是，它們也有嚴重的缺點。它們需要大量參數，這使得它們非常容易過適，有一個或兩個以上的隱藏層時變得難以訓練。在許多情況下，使用問題專用的非通用的模型可以獲得更好的結果。

本書的大部分內容討論在生命科學中特別有用的特定模型，後面的章節會介紹。但為了基本介紹的目的，我們應該討論兩種非常重要的模型，它們被廣泛用於許多不同的領域。它們被稱為卷積神經網路和遞歸神經網路。

卷積神經網路

卷積神經網路（*Convolutional Neural Network*，CNN）是最廣泛使用的深層模型之一，它們用於圖像處理和計算機視覺。它們仍然是在矩形網格上採樣的連續數據的多種問題的最佳選擇：聲音信號（1D）、圖像（2D）、立體 MRI 數據（3D）等。

它們也是一類真正表示“神經網路”的模型，CNN 的設計最初的靈感來自貓科視覺腦皮層的運作（貓從一開始就在深度學習中發揮了核心作用）。從 1950 年代到 1980 年代進行的研究顯示視覺是透過一系列的層處理，第一層中的每個神經元從視野的小區域（**其感受野**）獲取輸入。不同的神經元專門用於檢測特定的局部圖案或特徵，例如垂直或水平線。第二層中的單元從第一層中的局部單元簇中獲取輸入，將它們的信號組合以在較大的感受野上檢測更複雜的圖案。每個圖層都可以被視為原始圖像的新表示，用比前一層中更大和更抽象的圖案來描述。

CNN 反映了這種設計，以一系列層發送輸入圖像。從這個意義上講，它們就像 MLP 一樣，但每層的結構都非常不同。MLP 使用**完全連接層**，輸出向量的每個元素都取決於輸入向量的每個元素。CNN 使用空間局部性的**卷積層**，每個輸出元素對應於圖像的小區域，並且僅取決於該區域中的輸入值。這極大的減少定義每個層的參數數量。實際上，它假設權重矩陣 $\mathbf{M}_i$ 的大多數元素是 0，因為每個輸出元素僅依賴於少量輸入元素。

卷積層更進一步：它們假設參數對於**圖像的每個局部區域**都是相同的。如果圖層使用一組參數來檢測圖像中某個位置的水平線，則它也會使用完全相同的參數來檢測圖像中其他位置的水平線，這使得圖層的參數數量與圖像的大小無關。它需要學習的只是一個**卷積內核**，它定義如何從圖像的任何局部區域計算輸出特徵。該局部區域通常非常小，可能是 5 乘 5 像素。在這種情況下，要學習的參數數量僅為每個區域的輸出要素數量的 25 倍。與完全連接層中的數量相比，這是微不足道的，使得 CNN 更容易訓練且比 MLP 更不容易過適。

遞歸神經網路

遞歸神經網路（*Recurrent Neural Network*，RNN）略有不同。它們通常用於處理採用元素序列形式的數據：文件中的單詞、DNA 分子中的鹼基等。序列中的元素一次一個輸入到網路的輸入中，但是網路做了一些非常不同的事情：每一層的輸出在下一步被回饋到自己的輸入中！ 這讓 RNN 具有某種記憶。序列中的元素（單詞、DNA 鹼基等）被送入網路時，每層的輸入取決於該元素，但也取決於所有先前的元素（圖 2-4）。

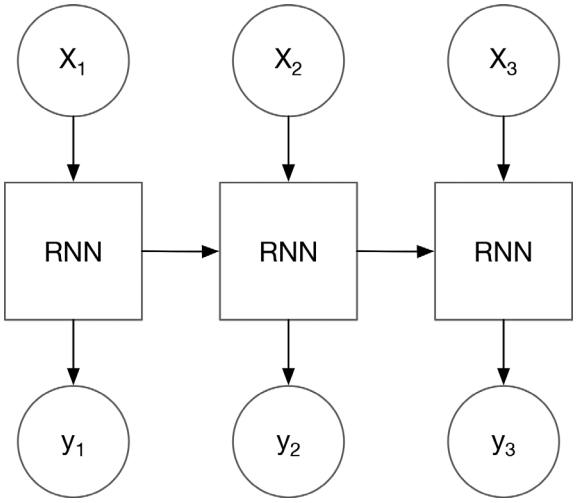

圖 2-4　遞歸神經網路。輸入序列的每個元素（x_1, x_2, ...），輸出（y_1, y_2, ...）視輸入元素與 RNN 本身先前步驟的輸出而定。

因此，遞歸層的輸入有兩部分：正規輸入（即網路中前一層的輸出）和遞歸輸入（等於上一步的輸出），然後它根據這些輸入計算新輸出。原則上你可以使用完全連接層，但在實務上通常不能很好的運作。研究人員開發了其他類型的層，這些層在 RNN 中的效果更好。兩個最受歡迎的層稱為**閘控遞歸單元**（*gated recurrent unit*，GRU）和**長短期記憶**（*long short-term memory*，LSTM）。現在不要擔心細節；只需記住建立 RNN 時通常應該使用其中一種類型的層來。

記憶能力使得 RNN 與我們討論過的其他模型有根本的不同。使用 CNN 或 MLP 只需將值輸入網路的輸入並獲得不同的值，輸出完全由輸入決定。RNN 並非如此，該模型有自己的內部狀態，由最近一步的所有層的輸出組成。每次將新值提供給模型時，輸出不僅取決於輸入值，還取決於內部狀態。同樣的，每個新輸入值都會改變內部狀態，這使得 RNN 非常強並能用於許多不同的應用程序。

推薦閱讀

深度學習是一個很大的課題，這一章做出最簡單的介紹應該足以幫助你閱讀和理解本書的其餘部分，但如果打算在這一領域認真發展，您需要更全面的背景知識。幸運的是，網路上有很多優秀的深度學習資源。以下是一些參考建議：

- *Neural Networks and Deep Learning*（*http://neuralnetworksanddeeplearning.com*），Michael Nielsen 著（Determination Press）討論與本章大致相同的內容，但詳細介紹了每個主題。如果你想深入深度學習的基礎知識並在工作中使用，這是一個很好的起點。
- Ian Goodfellow、Yoshua Bengio、Aaron Courville 的 *Deep Learning*（MIT Press，*http://www.deeplearningbook.org*）是該領域一些頂尖研究人員寫的更高階的介紹，這本書希望讀者有類似於計算機科學研究生的背景，並深入研究該主題背後的數學理論。你可以輕鬆使用深模型而無需理解所有理論，但如果想在深度學習中進行原創性研究（而不是僅僅使用深模型作為解決其他領域問題的工具），本書是一本很棒的資源。
- Bharath Ramsundar 和 Reza Zadeh 的 *TensorFlow for Deep Learning*（O'Reilly）提供了一個從業者對深度學習的介紹，旨在建立關於核心概念的直覺而不深入研究這些模型的數學基礎。對於那些對深度學習的實踐方面感興趣的人來說，這可能是一個有用的參考。（繁體中文版書名為《初探深度學習｜使用 *TensorFlow*》，碁峰資訊出版）