前言

歡迎來到《機器學習模擬應用》！這本書結合了您最喜歡的兩件事：電玩遊戲引擎還有人工智慧。我們希望您享受閱讀本書的程度就和我們享受寫作它的程度是一樣的。

具體來說，本書探討了 Unity 的使用，該產品曾經被稱為遊戲引擎（*game engine*），但現在更喜歡被稱為用於建立和操作互動式即時 *3D* 內容的平台。這句話很長，但基本上可以歸結為：Unity 是一個用於建構 3D 事物的平台，雖然它傳統上被用在電玩遊戲開發，但它可以用來建構任何可以用 3D 來表達的東西，藉由使用由 3D 圖形、物理模擬、和某種輸入所構成的組合。

藉由用於建立和操作互動式即時 *3D* 內容的平台與機器學習工具相結合，您可以使用所建立的 3D 世界來訓練機器學習模型，就好像它是真實世界（real-world）一樣。實際上它並不像真實世界，但想像起來會很有趣，並且與真實世界有一些合法有用的聯繫（例如能夠產生用於現實世界機器學習應用程式的資料，以及可以轉換為實體（physical）、真實世界的物件，例如機器人）。

當我們說真實世界時，實際上是指實體世界。

將 Unity 與機器學習相結合是建立模擬（*simulation*）和合成資料（*synthetic data*）的好方法，這將是本書所涵蓋的兩個不同主題。

本書使用的資源

我們建議您在閱讀每一章時都自己編寫程式碼來遵循本書的進度。

如果您遇到困難，或者只是想將我們版本的程式碼副本歸檔，可以透過我們的網站（*http://www.secretlab.com.au/books/practical-simulations*）找到您所需要的內容。

對於在本書中所完成的某些活動，您需要一份資源的副本才能獲得某些資產，因此我們建議您下載它。

讀者群和方法

我們為了對機器學習感興趣，但不一定是機器學習工程師的程式設計師和軟體工程師撰寫了這本書。如果您曾經對機器學習有興趣，或者開始在機器學習領域開展更多工作，那麼本書很適合您。如果您是遊戲開發者，已經了解了 Unity 或其他遊戲引擎，並且想要學習機器學習（無論是遊戲還是其他應用程式），那麼這本書也適合您閱讀。

如果您已經是機器學習專家，那麼本書也適合您，但方式不同：我們不會**太**深入探討機器學習的**原因**和**方法**。因此，如果您已經非常了解 PyTorch 和類似的框架的話，那麼您在這裡會如魚得水。如果您還不知道機器學習世界的內情，那也無妨，因為一切都**非常**容易獲得。使用 Unity 來進行模擬和合成的關鍵在於，您無需了解正在發生的事情的來龍去脈。它就是**有用**（這是著名的遺言，我們知道）。

無論您來自軟體、機器學習、或遊戲領域，這本書也適合您。這裡有適合所有人的東西。我們會教您剛好夠用的 Unity 和機器學習，我們也將為您提供起點，讓您能進一步了解您感興趣的路徑。

本書的組織

本書分為三個部分。

第一部分「模擬與合成的基礎知識」介紹了模擬與合成的主題，並透過每個主題的簡單活動來讓您輕鬆入門。

第二部分「模擬世界以獲得樂趣和利潤」是專門用在介紹模擬。這是本書最大的一部分，因為模擬是一個比合成還大得多的主題。在這一部分中，我們幾乎是一步一步地完

成了一系列模擬活動，同時還建構了額外的概念和方法。在本部分結束時，您將接觸到許多不同的模擬路徑。

第三部分「合成資料，真實結果」專門用在介紹合成。相較模擬來說這部分小得多，但仍然至關重要。您將學習如何使用 Unity 來建立合成資料的基礎知識，到最後您將有能力可以進行您可能會需要的任何類型的合成。

如何使用本書

我們環繞著活動來建立本書的結構。我們希望您能與我們一起完成這些活動，並在您喜歡的地方添加您自己的編撰（但請不要覺得您必須這樣做）。

本書採用了基於活動的方法，因為我們認為這是從 Unity 遊戲引擎和機器學習方面來學習所需內容的最佳方式。我們不想教您有關 Unity 的一切，而且本書也沒有足夠的篇幅來解開機器學習的所有細節。

透過從一個活動到另一個活動，我們可以根據需要來引入或排除事物。真心希望您喜歡我們選擇的活動！

我們的任務

對於模擬，我們將建構：

- 第 2 章，一個可以自己滾向目標的球（我們知道，這聽起來太不可思議了，但確實如此！）。
- 第 4 章，可以將方塊推入目標區域的立方體。
- 第 5 章，一輛簡單的自動駕駛汽車，可以在軌道上導航。
- 第 6 章，尋找硬幣的球，透過模仿人類的示範來進行訓練。
- 第 8 章，使用課程學習來建構可以向目標發射球的彈道發射器代理人。
- 第 9 章，一組立方體，它們可以協同工作來將方塊推向目標。
- 第 10 章，代理人可以使用視覺輸入（也就是相機），而不是精確的測量來平衡自身頂部的球。
- 第 11 章，用 Python 來連接和操作模擬的方法。

對於合成，我們將會：

- 第 3 章，產生會被隨機投擲和放置骰子影像。
- 第 13 章，改善骰子影像產生器，更改骰子的地板和顏色。
- 第 14 章，產生超市產品的影像，以允許對具有複雜背景和隨意定位的影像進行非 Unity 訓練。

本書使用慣例

本書使用以下印刷慣例：

斜體字（*Italic*）

表示新的術語、URL、電子郵件地址、檔名和延伸檔名。中文採用楷體字。

定寬字（`Constant width`）

用於程式列表，以及在段落中參照的程式元素，例如變數或函數名稱、資料庫、資料型別、環境變數、敘述和關鍵字。也用於命令和命令行輸出。

定寬粗體字（`Constant width bold`**）**

顯示命令或其他應由使用者輸入的文字。

定寬斜體字（*`Constant width italic`*）

顯示應該被使用者提供的值或根據語境（context）決定的值所取代的文字。

此圖示用來提出一個提示或建議。

此圖示用來提出一個一般性注意事項。

此圖示指出一個警告或警示事項。

第 1 章

合成與模擬介紹

這世界渴望著資料。機器學習和人工智慧是一些最需要資料的領域。演算法和模型越來越大，而真實世界的資料卻不夠充份。手動建立的資料和真實世界的系統是無法擴展的，所以我們需要新的方法。這就是 Unity 還有傳統上用於電玩遊戲開發的軟體可以介入的地方。

本書完全和合成與模擬有關，以及利用現代電玩遊戲引擎的力量來進行機器學習。將機器學習與模擬以及合成資料相結合這件事，表面上聽起來還頗為簡單，但現實是，將**電玩遊戲**技術納入機器學習的嚴肅商業世界這樣的想法，嚇跑了超乎想像多的公司和企業。

我們希望這本書能引導您進入這個世界並減輕您的擔憂。本書的三位作者是具有重要電腦科學背景的電玩遊戲開發人員，還有一位是認真的機器學習和資料科學家。我們多年來在各種行業和方法中所建立的綜合觀點和知識，將在這裡為您呈現。

本書將帶您了解可用於建構和訓練機器學習系統的方法和技術，並使用由 Unity 電玩遊戲引擎所產生的資料。本書有兩個不同的領域：**模擬**（*simulation*）和**合成**（*synthesis*）。對所有的意圖與目的而言，模擬是指建構會學習在您自己建立的虛擬世界中做某件事的虛擬機器人（稱為**代理人**（*agent*））。合成是指建構虛擬物件或世界、輸出有關這些物件和世界的資料、並使用它來訓練遊戲引擎之外的機器學習系統。

模擬和合成都是強大的技術，可以為以資料為中心的機器學習和人工智慧提供令人興奮的新方法。

ML 的嶄新世界

我們很快就會了解本書的結構，但首先，以下是本章概要，分為四個部分：

- 在「領域」小節中，我們將介紹本書探索的機器學習領域：模擬和合成。
- 在第 6 頁的「工具」中，我們將了解使用的工具（*Unity 引擎*、*Unity ML-Agents Toolkit*、*PyTorch*、以及 *Unity Perception*）還有它們如何是組合在一起。
- 在第 9 頁的「技術」中，我們將了解用於機器學習的技術：**近端策略優化**（*proximal policy optimization*, PPO）、**柔性演員 - 評論家**（*soft actor-critic*, SAC）、**行為複製**（*behavioral cloning*, BC）、和**生成對抗模仿學習**（*generative adversarial imitation learning*, GAIL）。
- 最後，在第 13 頁的「專案」中，我們將總結本書中所建構的專案，以及它們與領域和工具的關係。

本章結束時，您將準備好深入模擬和合成的世界，並大致了解遊戲引擎的工作原理，並且您會明白為什麼它是近乎完美的機器學習工具。在本書的最後，您將準備好解決您能想到的任何可以從由遊戲引擎驅動的模擬或合成中受益的問題。

領域

本書的兩大支柱是**模擬**和**合成**。在本節中，我們將準確解釋每個術語的涵義以及本書將如何探索這些概念。

模擬和合成是人工智慧和機器學習未來的核心部分。

許多應用將立即出現在您眼前：將模擬與深度強化學習相結合，以在建構實體產品之前先驗證新機器人的功能；在沒有汽車的情況下建立自動駕駛汽車的大腦；建立您的倉庫並在沒有倉庫（或機器人）的情況下訓練您的取放型機器人（pick-and-place robot）。

其他用途則更為微妙：使用模擬來合成資料以建立人工資料，而不是從真實世界紀錄的資訊，然後用來訓練傳統的機器學習模型；獲取真實的使用者活動，並結合行為複製和模擬，使用它為原本完美的機器學習任務添加生物或人類外觀的元素。

諸如 Unity 之類的電玩遊戲引擎可以以足夠的依國家教育研究院樂詞網使用傳真度（fidelity）來模擬足夠多的真實世界，從而可用於基於模擬的機器學習和人工智慧。遊

戲引擎不僅可以讓您模擬足夠多的城市和汽車來測試、訓練、和驗證自動駕駛汽車的深度學習模型，還可以模擬硬體到引擎溫度、剩餘電力、光達（LIDAR）、聲納、X 光等程度。想在您的機器人中加入一個花俏、昂貴的新感測器嗎？在您投資新設備之前，請先試一試，看看它是否可以提高效能。節省金錢、時間、計算能力、和工程資源，並更了解您的問題空間。

獲取足夠的資料真的是不可能，或者不安全嗎？建立一個模擬並測試您的理論吧。廉價、無限的訓練資料終究只是一個模擬而已。

模擬

當我們說著*模擬*時，所指的不是一件具體的事情。在這種情況下，模擬實際上意味著使用遊戲引擎來開發隨後將應用機器學習的場景或環境。在本書中，我們使用模擬作為一個術語來泛指以下內容：

- 使用遊戲引擎來建立具有特定組件的環境，這些組件是一個代理人或一組代理人。
- 賦予代理人移動環境和 / 或其他代理人的能力，或以其他方式與環境和 / 或其他代理人互動或一起工作。
- 將環境連接到機器學習框架以訓練可以在環境中操作代理人的模型。
- 使用經過訓練的模型在未來與環境一起進行操作，或將模型連接到其他具有類似配備的代理人（例如，在真實世界中，使用真實的機器人）。

合成

合成是一件容易確定的事情：在本書的語境（context）中，**合成**就是使用遊戲引擎來建立看來虛假的訓練資料。例如，如果您正在為超市建構某種影像識別機器學習模型，您可能需要從許多不同的角度以及在許多不同的背景和語境下拍攝一盒特定穀片品牌的照片。

使用遊戲引擎，您可以建立和載入一盒穀片的 3D 模型，然後以不同的角度、背景和傾斜度產生數千張影像（合成它們），並將它們儲存為標準影像格式（例如 JPG 或 PNG）。然後，憑藉大量訓練資料，您可以使用完美標準的機器學習框架和工具箱（例如 TensorFlow、PyTorch、Create ML、Turi Create 或眾多基於 Web 服務的訓練系統之一）並訓練模型來識別您的穀片盒。

然後可以將這種模式部署到，例如，某種手推車上的人工智慧系統，幫助人們購物、引導他們到購物清單上的物品、或者幫助商店員工正確地裝滿貨架並進行庫存預測。

合成就是使用遊戲引擎建立訓練資料，而遊戲引擎通常與訓練過程本身沒有任何關係或只有很少的關係。

工具

本章向您介紹我們將在旅程中使用的工具。如果您不是遊戲開發人員，您將遇到的主要新工具是 Unity。Unity 傳統上是一個遊戲引擎，但現在被稱為即時 3D 引擎。

讓我們一一介紹您將在本書中遇到的工具。

Unity

首先，Unity 是一個遊戲和視覺效果引擎。Unity Technologies 將 Unity 描述為一個*即時 3D 開發平台*。我們不會為您重複 Unity 網站上的行銷素材，但如果您對該公司是如何定位自己感到好奇的話，可以查看他們的網站（*https://oreil.ly/nnVUz*）。

本書不是來講述 Unity 的基礎知識。這本書的一些作者已經寫了幾本關於這方面的書（從遊戲開發的角度來看）如果您有興趣的話，可以在 O'Reilly Media 找到這些書。作為遊戲開發人員，您無需學習 Unity 即可使用它來進行機器學習的模擬和合成；在本書中，我們將教您剛好夠用的 *Unity* 來有效地完成這件事。

Unity 使用者介面看起來和幾乎所有其他具有 3D 功能的專業軟體套件一樣。我們在圖 1-1 中包含了一個範例螢幕截圖。該介面具有可操作的窗格、用於處理物件的 3D 畫布以及許多設定（setting）。稍後我們將回到 Unity 使用者介面的細節。

您可以在 Unity 說明文件（*https://oreil.ly/zN8xU*）中全面了解它的不同元素。

在本書中，您將使用 Unity 來進行模擬和合成。

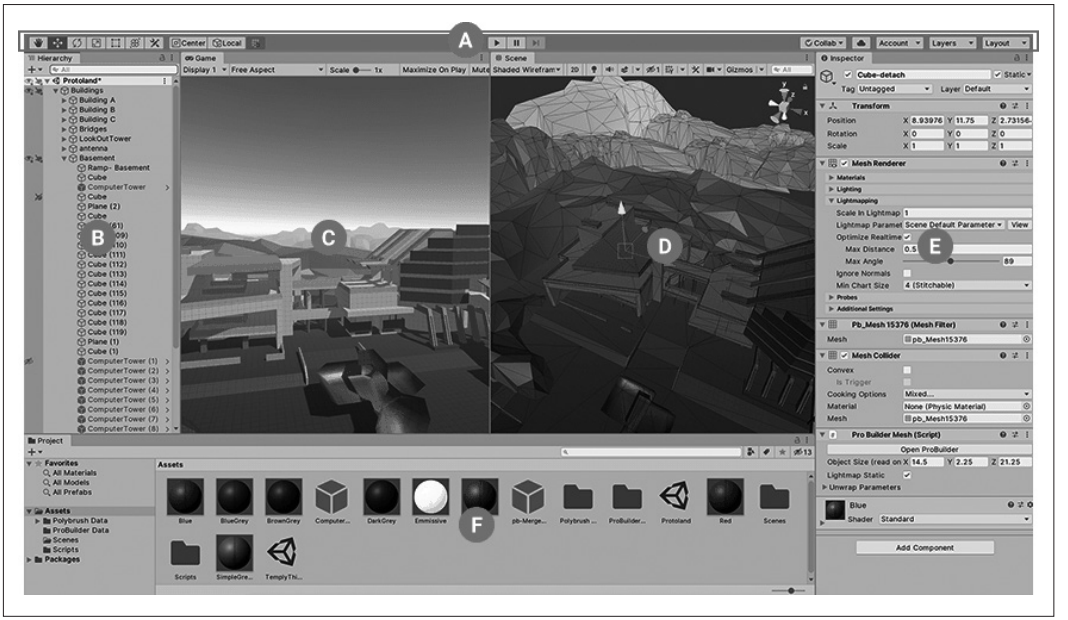

圖 1-1　Unity 使用者介面

Unity 引擎帶有一組強大的工具，可讓您模擬重力、力、摩擦、運動、各種感測器等。這些工具正是建構現代電玩遊戲所需的工具集。事實證明，這些也是建立模擬和合成機器學習資料所需的工具集。但鑑於您正在閱讀本書，您可能已經猜到這件事了。

這本書是為 Unity 2021 以及更高版本編寫的。如果您在 2023 年或之後閱讀本書，Unity 可能看起來與我們的螢幕截圖略有不同，但概念和整體流程應該不會有太大變化。總體而言，遊戲引擎傾向於會累積功能而不是刪除，因此您會看到最常見的變化類型是圖示看起來會略有不同以及類似性質的事情。有關可能已更改任何內容的最新說明，請訪問我們為本書所設的專門網站（*https://oreil.ly/1efRA*）。

透過 Unity ML-Agents 的 PyTorch

如果您在機器學習領域，可能聽說過 PyTorch 開源專案。作為學術界和產業界最受歡迎的機器學習平台和生態系統之一，它幾乎無處不在。在模擬和合成領域，事情沒有什麼不同：PyTorch 還是首選框架之一。

在本書中，我們所探索的底層機器學習將主要透過 PyTorch 來完成。我們不會深入研究 PyTorch，因為我們會使用 PyTorch 來進行的大部分工作，並將會透過 Unity ML-Agents Toolkit 來進行。我們很快就會討論 ML-Agents Toolkit，但基本上您需要記住的是 PyTorch 是推動 Unity ML-Agents Toolkit 所做工作的**引擎**。它一直都在引擎蓋下，所以如果需要，或者您知道在做什麼時，就可以修改它，但大多數時候您根本不需要碰觸它。

我們將在本節的剩餘部分討論 Unity ML-Agents Toolkit，因此如果您需要複習 PyTorch，我們強烈推薦 PyTorch 網站（*https://pytorch.org*），或 O'Reilly Media 出版的有關該主題的眾多優秀書籍。

PyTorch 是一個支援使用資料流圖（data flow graph）來執行計算的程式庫。它支援使用 CPU 和 GPU（以及其他專門的機器學習硬體）來進行訓練和推理，並且可以在強大的 ML 優化伺服器到行動裝置的各種平台上運行。

因為在本書中您將使用 PyTorch 所進行的大部分工作都是抽象化的，所以我們很少會談論 PyTorch 本身。因此，雖然它幾乎是我們將要探索的所有內容的背景工具，但您使用它的主要介面將是透過 Unity ML-Agents Toolkit 和其他工具。

我們將透過 Unity ML-Agents 來使用 PyTorch 以進行本書中的所有模擬活動。

Unity ML-Agents Toolkit

Unity ML-Agents Toolkit（相對於 Unity 品牌，我們很多時候將它縮寫為 *UnityML* 或 *ML-Agents*）是您將在本書中進行的工作支柱。ML-Agents 最初是作為一個簡單的實驗性專案發布，並逐漸發展為包含一系列功能，使 Unity 引擎能用作訓練和探索智慧型代理人和其他機器學習應用程式的模擬環境。

這是一個開源專案，附帶了許多令人興奮且經過深思熟慮的範例（如圖 1-2 所示），並且可以透過其 GitHub 專案（*https://oreil.ly/JPkQ8*）免費獲得。

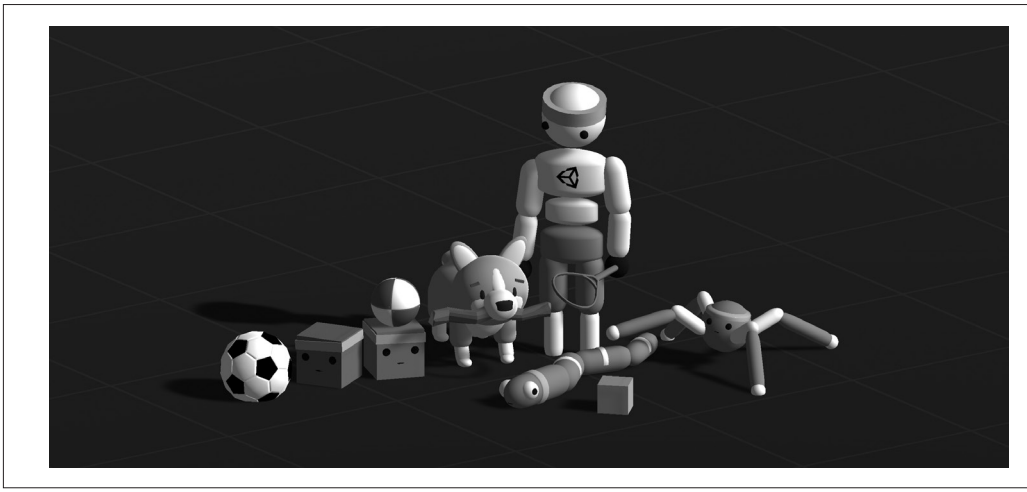

圖 1-2　Unity ML-Agents Toolkit 的「英雄影像」，展示了 Unity 的一些範例角色

如果這還不夠明顯，我們將在本書的所有模擬活動中使用 ML-Agents。並且將在第 2 章中向您展示如何在您自己的系統上啟動和執行 ML-Agent。不要急於安裝它！

Unity Perception

Unity Perception 套件（我們在很多時候將其縮寫為 *Perception*）是我們將用來產生合成資料的工具。Unity Perception 為 Unity Editor 提供了一系列附加功能，允許您適當地設定場景以建立假資料。

和 ML-Agents 一樣，Perception 是一個開源專案，您可以透過其 GitHub 專案（*https://oreil.ly/KbvHj*）來找到它。

技術

ML-Agents Toolkit 支援使用**強化學習**（*reinforcement learning*）和**模仿學習**（*imitation learning*）技術中的其中一種或其組合來進行訓練。它們每一個都允許代理人透過反復的嘗試錯誤（或「強化」）來「學習」所需的行為，並最終收斂在符合所提供的成功準則的理想行為。這些技術之間的不同之處在於準則，這些準則將用於評估和優化代理人的效能。

強化學習

強化學習（RL）是指採用外顯式獎勵的學習過程。由實作來決定為可取的行為獎勵「積分」，並為不受歡迎的行為扣分。

此時您可能會想，**如果我還必須告訴它什麼該做、什麼不該做，那麼機器學習的意義何在？**但是，讓我們以教導雙足代理人走路為例。為了步行所需的每個狀態變化而給出的一組明確指令（也就是每個關節應該依順序旋轉的確切程度）將會是大量且複雜的。

但是，透過在代理人往終點線移動時給幾分、到達終點時給很多分、跌倒時給負分、還有數十萬次的嘗試來讓它正確移動，它將能夠自己弄清楚其中的細節。因此，RL 的強大之處在於能夠給出以目標為中心的指令，而這需要複雜的行為才能達成。

ML-Agents 框架附帶了兩種不同的內建 RL 演算法的實作：**近端策略優化**（*proximal policy optimization*, PPO）和**柔性演員 - 評論家**（*soft actor-critic*, SAC）。

請注意這些技術和演算法的首字母縮寫詞：RL、PPO 和 SAC。請記住它們。我們將在本書中經常使用。

PPO 是一種功能強大的泛用 RL 演算法，已被反復證明在一系列應用程式中非常有效且通常很穩定。PPO 是 ML-Agents 中使用的預設演算法，本書的大部分內容都將使用它。稍後我們將更詳細地探討 PPO 的工作原理。

近端策略優化由 OpenAI 團隊建立並於 2017 年首次亮相。如果您有興趣深入了解其中細節，可以閱讀 arXiv（*https://oreil.ly/JHfhI*）上的原始論文。

SAC 是一種**離策略**（*off-policy*）的 RL 演算法。我們稍後會了解這意味著什麼，但就目前而言，它通常會減少所需的訓練週期數，代價是增加記憶體需求。與 PPO 等**同策略**（*on-policy*）方法相比，這使其成為慢速訓練環境的更好選擇。我們將在本書中使用一次或兩次的 SAC，當我們談論到時，將更詳細地探討它是如何運作的。

柔性演員 - 評論家由伯克萊人工智慧研究（Berkeley Artificial Intelligence Research, BAIR）小組建立，於 2018 年 12 月首次亮相。有關它的詳細資訊，您可以閱讀原始發行版本的說明文件（*https://oreil.ly/7kNmg*）。

模仿學習

和 RL 類似，**模仿學習**（*imitation learning*, IL）免除了定義複雜指令的需要，有利於簡單地設定目標。然而，IL 也免除了定義明確目標或獎勵的需要。取而代之的是，給出了一個示範（通常是一個由人類手動控制的代理人的紀錄）並且獎勵**本質上**是基於代理人**模仿**所示範的行為來定義的。

這對於那些期望行為是非常具體的，或者絕大多數可能的行為都是不受期望的複雜領域非常有用。使用 IL 來進行訓練對於多階段目標也非常有效 —— 代理人需要以特定順序來達成中間目標才能獲得獎勵。

ML-Agent 框架附帶了兩種不同的內建 IL 演算法的實作：**行為複製**（*behavioral cloning*, BC）和**生成對抗模仿學習**（*generative adversarial imitation learning*, GAIL）。

BC 是一種 IL 演算法，可以訓練代理人去精確模仿所示範的行為。在這裡，BC 只負責定義和分配內在獎勵；既有的 RL 方法（例如 PPO 或 SAC）被用於其下的訓練過程。

GAIL 是一種生成對抗方法，適用於 IL。在 GAIL 中，兩個獨立的模型在訓練過程中相互競爭：一個是代理人行為模型，它會盡力去模仿所給定的示範；另一個是鑑別器（discriminator），它反復地扮演由人類驅動的示範者行為片段，或由代理人驅動的模型行為片段，並且必須猜測它是其中哪一個。

GAIL 起源於 Jonathan Ho 和 Stefano Ermon 的論文「Generative Adversarial Imitation Learning」（*https://oreil.ly/bokpR*）。

隨著鑑別器在識別模仿者這方面變得更好，代理人模型必須改進才能再次欺騙它。同樣的，隨著代理人模型的改進，鑑別器必須建立越來越嚴格或細緻入微的內部準則來識別假貨。在這種來回中，它們都被迫進行迭代式的改進。

行為複製通常是使用在應用的最佳方法，因為在這些應用中可以示範出代理人所有（或幾乎所有）可能發現的條件。相反的，GAIL 能夠推斷出新的行為，從而可以從有限的示範中學習模仿。

BC 和 GAIL 也可以一起使用，通常是在早期訓練中使用 BC，然後將部分的訓練行為模型配置為 GAIL 模型中代理人的那一半。從 BC 來開始通常會使代理人在早期訓練中快速改善，而在後期訓練中切換到 GAIL 將允許它發展超出所示範的行為。

混合學習

儘管單獨使用 RL 或 IL 幾乎總是能解決問題，但它們也可以結合使用。然後可以透過達成目標時的外顯式定義的獎勵還有進行有效模仿時的內隱式獎勵來獎勵代理人及其行為。兩者的權重甚至可以調整，以便可以訓練代理人將其中之一作為主要目標或兩者皆作為同等重要的目標。

在混合訓練中，IL 示範有助於在訓練早期將代理人置於正確的路徑上，而外顯式的 RL 獎勵則鼓勵在此路徑內或之外的特定行為。這件事在理想代理人應該要能夠勝過人類示範者的那些領域中是必要的。由於早期的手把手訓練，同時使用 RL 和 IL 進行訓練可以顯著地加快訓練代理人來解決複雜問題，或在獎勵稀少的場景中對複雜環境進行導航的速度。

稀疏獎勵環境（*sparse-reward environment*）就是那些代理人會獲得外顯式獎勵特別少的環境。在這樣的環境中，代理人會「意外地」偶然發現一個值得獎勵的行為（並因此收到它應該做什麼的第一個指示）所花費的時間可能會浪費大量可用的訓練時間。但結合了 IL 之後，該示範可以告知可獲得外顯式獎勵的理想行為。

這些方法共同產生了一個複雜的獎勵方案，可以鼓勵代理人進行高度特定的行為，但是需要如此複雜等級才能使代理人成功的應用並不多。

技術總結

本章是對概念和技術的介紹性總覽，在本書課程中您將接觸和使用我們在這裡所看到的每一種技術。在這樣做的過程中，您將更加熟悉它們每一個在實際意義上是如何工作的。

要點如下：

- Unity ML-Agents Toolkit 目前提供了兩個類別的一系列訓練演算法：
  - 對於強化學習（RL）：近端策略優化（PPO）和柔性演員 - 評論家（SAC）
  - 對於模仿學習（IL）：行為複製（BC）和生成對抗模仿學習（GAIL）
- 這些方法可以單獨使用或一起使用：
  - RL 可以單獨與 PPO 或 SAC 一起使用，也可以與 IL 方法（如 BC）結合使用。
  - BC 可以作為使用 GAIL 的方法的一個步驟來單獨使用，或與 RL 結合使用。
- RL 技術需要一組已定義的獎勵。
- IL 技術需要某種形式的示範。
- RL 和 IL 都是在做中學（*learn by doing*）。

在本書對模擬主題其餘部分的探索中，我們將會接觸到或直接使用所有這些技術。

專案

本書是一本實用、務實的作品。我們希望您儘快啟動並執行模擬和合成，並且我們假設您希望儘可能聚焦於實作。

因此，雖然我們經常在探索背後的真相，但本書的精髓在於我們將共同建構的專案。

本書中實用的、基於專案的那方面區分成之前討論的兩個領域：模擬和合成。

模擬專案

我們的模擬專案將是各式各樣的：當您在 Unity 中建構模擬環境時，存在於環境中的代理人可以透過多種方式來觀察和感知其世界。

有一些模擬專案將會使用一個使用了向量觀察值（*vector observation*）來觀察世界的代理人：向量觀察值也就是數字。不論您想送出的是什麼數字。從字面上看，也就是任何您喜歡的東西。實際上，向量觀察值通常像是代理人與某物的距離或其他位置資訊之類的東西。但實際上，任何數字都可以是一個觀察值。

有一些模擬專案將使用透過**視覺觀察值**（*visual observation*）（也就是圖片！）來觀察世界的代理人。因為 Unity 是一個遊戲引擎，而遊戲引擎和電影一樣，都有**相機**（*camera*）的概念，所以您可以簡單地在您的代理人上（虛擬地）安裝相機，並讓它存在於遊戲世界中。然後可以將來自這些相機的視圖輸入到您的機器學習系統中，讓代理人根據相機的輸入來了解其世界。

我們將使用 Unity、ML-Agents、和 PyTorch 來進行的模擬範例包括：

- 第 2 章，一個可以自己滾向目標的球（我們知道，這聽起來太不可思議了，但確實如此！）。
- 第 4 章，可以將方塊推入目標區域的立方體。
- 第 5 章中一輛可以在軌道上行駛的簡單自動駕駛汽車。
- 第 6 章，尋找硬幣的球，透過模仿人類的示範來進行訓練。
- 第 8 章，一種彈道發射器代理人，可以使用課程學習（curriculum learning）向目標發射球。
- 第 9 章，一組立方體，它們會協同工作將方塊推向目標。
- 第 10 章，訓練代理人使用視覺輸入（也就是相機）而不是精確測量地將球平衡在自己的上面。
- 第 11 章，使用 Python 來連接和操作 ML-Agent。

合成專案

我們的合成專案將比我們的模擬專案還少，因為這領域相對更簡單一些。我們專注於利用 Unity 所提供的素材來展示模擬的可能性。

我們將使用 Unity 和 Perception 來進行的合成範例包括：

- 第 3 章，使用隨機投擲和放置的骰子的影像產生器。
- 第 13 章，透過改變骰子的地板和顏色來改進骰子影像產生器。
- 第 14 章，產生超市產品的影像以允許對具有複雜背景和隨意定位的影像進行非 Unity 訓練。

一旦您產生了合成資料，我們就不會再聚焦於實際的訓練過程，因為有很多很多關於這個主題的好書和線上貼文，本書就不再贅述。

總結和後續步驟

您已經邁出了第一步，本章包含了一些必需的背景素材。從這裡開始，我們將透過實踐來教您。這本書的標題中有「實用」一詞是有原因的，我們希望您藉由建構自己的專案來感受模擬和合成。

您可以在本書的專門網站（*https://oreil.ly/1efRA*）上找到每個範例的程式碼 —— 建議您只在必要時才去下載程式碼。我們還將根據您應該需要注意的任何變更來使網站保持最新狀態，因此請將它加入您的書籤！

下一章中，我們將了解如何建立您的第一個模擬，實作一個代理人以在其中做某件事，以及使用強化學習來訓練一個機器學習系統。