增强学习导论

作者：Rich Sutton 译者：Jackokie Zhao

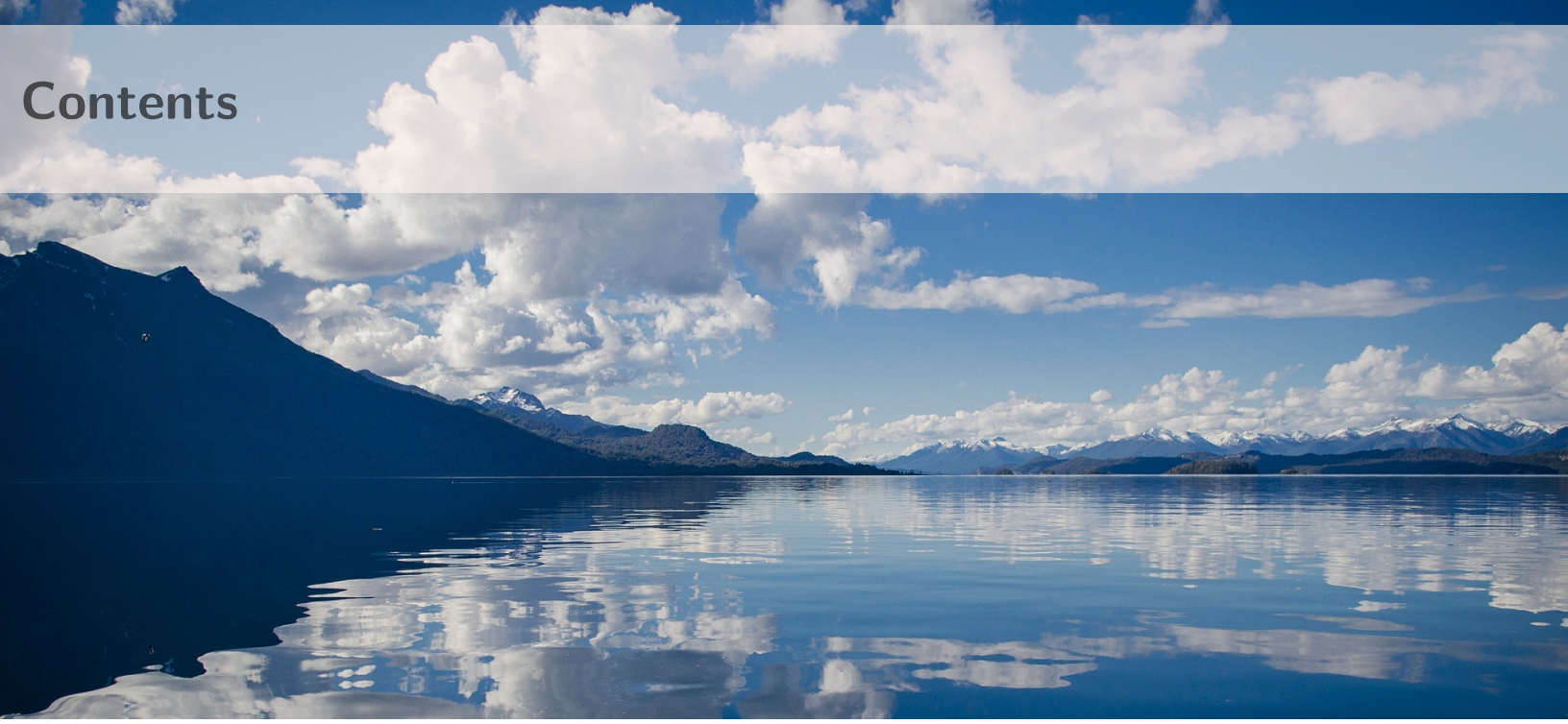
中国教育出版社

Art. No xxxxx
ISBN xxx–xx–xxxx–xx–x
Edition 0.0

Cover design by 赵纪伟

Published by 中国教育出版社
Printed in 南京

Contents



1. 第一章引言

当我们思考学习的本质时，我们首先想到的可能就是通过与环境的互动来学习。当婴儿玩耍、挥动手臂或四处张望时，它没有明确的老师，但它确实与环境有直接的感觉运动联系。这种联系产生了大量关于因果关系、行为后果以及为了实现目标该做什么的信息。在我们的一生中，这样的互动无疑是了解我们的环境和我们自己的主要来源。无论我们是在学习开车还是在交谈，我们都清楚地意识到我们的环境是如何回应我们的行为的，我们试图通过我们的行为来影响我们的行为。从互动中学习是几乎所有学习和智力理论的基础。在这本书中，我们探索了一种从交互中学习的计算方法。我们主要探讨的是理想化的学习情况，并评估各种学习方法的有效性，而不是直接理论化人们或动物是如何学习的。也就是说，我们采用人工智能研究者或工程师的视角。我们通过数学分析或计算实验来评估设计，来探索能够有效解决科学或经济问题的机器的设计。我们所探索的方法，叫做强化学习，比机器学习的其他方法更专注于目标导向的交互学习。

1.1 强化学习

强化学习是学习做什么——如何将情况映射到行动——从而最大化一个数字奖励信号。学习者并没有被告知要采取哪些行动，而是必须通过尝试来发现哪些行为可以获得最大的回报。在最有趣和最具挑战性的情况下，行动可能不仅影响眼前的事情

心理学和神经科学的关系在第 14 章和第 15 章总结。

奖励，但也包括下一个情境，通过这个，所有后续的奖励。这两个特征——试错搜索和延迟奖励——是强化学习的两个最重要的显著特征。强化学习，就像许多以“ing”结尾的主题一样，比如机器学习和登山，同时也是一个问题，是一种很好的解决问题的方法，也是研究这个问题及其解决方法的领域。对于这三件事使用一个名字是很方便的，但同时也必须保持这三件事在概念上是分开的。特别是在强化学习中，问题与解决方法的区别是非常重要的; 没有区分这一点是许多困惑的根源。我们利用动力系统理论中的思想，具体地说，将强化学习问题形式化为不完全已知马尔可夫决策过程的最优控制。这种形式化的细节必须等到第 3 章，但是基本的思想仅仅是捕获学习代理与它的环境在一段时间内交互的真实问题的最重要的方面来实现目标。学习代理必须能够在一定程度上感知其环境的状态，并且必须能够采取影响状态的行动。代理还必须有一个与环境状态相关的目标或目标。马尔可夫决策过程旨在将这三个方面——感觉、行动和目标——以它们最简单的可能形式包含进来，而不忽略它们中的任何一个。任何适合解决此类问题的方法我们都认为是一种强化学习方法。强化学习不同于监督学习，监督学习是目前机器学习领域中研究最多的一种学习。监督学习是指从一组由有知识的外部主管提供的标记示例中学习。每个示例都是对一种情况的描述，以及系统对这种情况应采取的正确行动的说明 (标记)，这通常是为了确定该情况所属的类别。这种学习的目

的是让系统对其反应进行外推或归纳，使其在训练集中不存在的情况下能够正确地发挥作用，这是一种重要的学习方式，但单独从交互中学习是不够的。在交互问题中，获取期望行为的实例通常是不现实的，这些行为既正确又能代表代理必须采取的所有情况。在未知领域——人们期望学习成为最有利的领域——一个代理人必须能够从自己的经验中学习。强化学习也不同于机器学习研究者所称的无监督学习，无监督学习通常是指寻找隐藏在无标记数据集合中的结构。“监督学习”和“无监督学习”这两个术语似乎会对机器学习范式进行彻底的分类，但它们并非如此。虽然人们可能会认为强化学习是一种无监督学习，因为它不依赖于正确行为的例子，强化学习是试图最大化奖励信号而不是试图寻找隐藏的结构。在 agent 的经验中发现结构在强化学习中肯定是有用的，但是它本身并不能解决激励信号最大化的强化学习问题。因此，我们认为强化学习是第三种机器学习范式，除了监督学习和非监督学习，也许还有其他范式。

强化学习的挑战之一，而不是其他类型的学习，是探索和开发之间的权衡。为了获得大量的奖励，强化学习代理必须更喜欢它过去尝试过并且发现在产生奖励方面有效的行为。但是要发现这样的行为，它必须尝试以前没有选择过的行为。为了获得奖励，行为人必须挖掘自己已经经历过的东西，但为了在未来做出更好的行动选择，行为人也必须进行探索。现在的困境是，无论是勘探还是开发都不能在不失败的情况下进行。代理必须尝试各种各样的操作，并逐渐偏爱那些看起来最好的。在随机任务中，每个动作都必须反复尝试，以获得对预期回报的可靠估计。摘要探索-开发困境已被数学家研究了几十年，但仍未得到解决。目前，我们只是注意到，在监督和非监督学习中，平衡勘探和开发的整个问题甚至没有出现，至少在这些范例的最纯粹的形式中。强化学习的另一个关键特性是，它明确地考虑了目标导向的代理与不确定环境交互的整个问题。这与许多考虑子问题的方法不同，这些方法没有解决如何将子问题融入更大的范围。例如，我们已经提到，许多机器学习研究涉及监督学习，但没有明确说明这种能力最终如何有用。其他研究人员已经开发出了具有一般目标的规划理论，但不考虑计划在实时决策中的作用，也不考虑规划所需要的预测模型从何而来的问题。虽然这些方法产生了许多有用的结果，但是它们对孤立子问题的关注是一个很大的限制。强化学习采取了相反的策略，从一个完整的、交互式的、目标寻求的代理开始。所有增强学习代理都有明确的目标，能够感知环境的各个方面，并且可以选择影响环境的行为。此外，通常从一开始就假定代理必须操作，尽管它面临的环境存在很大的不确定性。当强化学习涉及到规划时，它必须解决规划和实时行动选择之间的相互作用，以及如何获得和改进环境模型的问题。当强化学习涉及到监督学习时，它这样做是因为特定的原因，这些原因决定了哪些能力是关键的，哪些不是。为了学习研究取得进展，重要的子问题必须被隔离和研究，但它们应该是在完整的、交互的、目标寻求的代理中起着明确作用的子问题，即使完整代理的所有细节还不能被填满。我们所说的完全的、互动的、寻求目标的代理，并不总是指一个完整的有机体或机器人。这些显然是例子，但一个完整的、交互式的、目标寻求代理也可以是一个大型行为系统的组成部分。在这种情况下，代理直接与大型系统的其余部分交互，并间接地与大型系统的环境交互。一个简单的例子是一个代理，它监视机器人电池的充电水平，并向机器人的控制架构发送命令。这个 agent 的环境是机器人的其余部分以及机器人的环境。我们必须超越最明显的代理及其环境示例

欣赏强化学习框架的通用性。现代强化学习最令人兴奋的一个方面是它与其他工程和科学学科之间的实质性和富有成效的互动。强化学习是人工智能和机器学习几十年发展趋势的一部分，这一趋势是为了更好地与统计、优化和其他数学科目相结合。例如，一些强化学习方法能够用参数化逼近器进行学习，这就解决了运筹学和控制理论中经典的“维数诅咒”问题。更明显的是，强化学习也与心理学和神经科学有着强烈的相互作用，两者都带来了巨大的好处。在机器学习的所有形式中，强化学习是最接近人类和其他动物的学习方式，许多强化学习的核心算法最初是受到生物学习系统的启发。强化学习也有回报，既通过动物学习的心理模型更好地匹配一些经验数据，也通过大脑奖励系统的部分有影响力的模型。这本书的主体发展与工程和人工智能有关的强化学习的思想，与心理学和神经科学的联系在第 14 和 15 章总结。最后，强化学习也是人工智能回归简单的一般原则的更大趋势的一部分。自 20 世纪

60 年代末以来，许多人工智能研究人员认为没有普遍的原则可以被发现，取而代之的是由于拥有大量的特殊目的的诡计、程序和启发式。有时人们会说，如果我们能把足够的相关信息输入一台机器，比如一百万或十亿，那么它就会变得智能。基于一般原则 (如搜索或学习) 的方法被描述为“弱方法”，而基于特定知识的方法被称为“强方法”。这种观点在今天仍然很普遍，但并不占主导地位。从我们的观点来看，这完全是不成熟的: 在寻求一般原则时，几乎没有付出什么努力来得出根本没有原则的结论。现代人工智能现在包括很多研究，寻找学习、搜索和决策的一般原理。目前尚不清楚这个钟摆将会摆动多远，但强化学习研究肯定是朝着更简单、更少的人工智能一般原则的方向倒退的一部分。

1.2 强化学习的例子

理解强化学习的一个好方法是考虑一些指导它发展的例子和可能的应用。

国际象棋大师棋步。选择是通过计划——预测可能的回答和反驳——以及直接的、直觉的判断特定职位和行动的可取性来通知的。

- 自适应控制器实时调整炼油厂运行参数。控制器在指定的边际成本的基础上优化产量/成本/质量的权衡，而不严格遵循工程师最初建议的设置点。

1.3 强化学习的要素

一只小羚羊出生后几分钟就挣扎着站起来。半小时后，它以每小时 20 英里的速度运行。

- 一个移动机器人决定是进入一个新的房间寻找更多的垃圾来收集，还是开始寻找返回电池充电站的方法。它根据电池当前的充电水平以及过去找到充电器的速度和容易程度做出决定。

菲尔准备早餐。仔细研究后，即使是这种看似平凡的活动，也揭示了一个由条件行为和相互关联的目标-子目标关系组成的复杂网络: 走向橱柜，打开它，选择一个麦片盒子，然后伸手去拿，抓取，然后取回盒子。要得到一个碗、勺子和牛奶盒，还需要其他复杂的、经过调整的、相互作用的行为序列。每一步都需要一系列的眼球运动来获取信息，并引导接触和移动。人们不断做出快速的判断，比如如何携带这些物品，或者在获得其他物品之前，最好先将其中一些物品运送到餐桌上。每一步都有自己的目标，比如抓一把勺子或者去冰箱拿东西，这些目标都是为了实现其他的目标，比如在准备好麦片的时候用勺子吃饭，最终获得营养。不管他是否意识到这一点，菲尔正在获取关于他身体状况的信息，这些信息决定了他的营养需求、饥饿程度和食物偏好。

这些示例共享非常基本的特性，因此很容易被忽略。所有这些都涉及到一个活跃的决策代理和它的环境之间的交互，在这些交互中，代理试图在不确定的环境下实现目标。agent 的行为被允许影响环境的未来状态 (例如，下一个象棋位置，炼油厂的储藏量水平，机器人的下一个位置和电池的未来充电水平)，从而影响 agent 在以后的行为和机会。正确的选择需要考虑到行动的间接、延迟的后果，因此可能需要远见或计划。与此同时，在所有这些例子中，行动的影响不能完全预测; 因此，代理必须频繁地监视它的环境并作出适当的反应。例如，菲尔必须注意他倒进麦片碗里的牛奶，以防止牛奶溢出来。所有这些示例都涉及明确的目标，因为代理可以根据其能够直接感知的内容来判断实现目标的进展。棋手知道自己是否赢了，炼油厂的控制器知道生产了多少石油，羚羊知道石油何时坠落，移动机器人知道电池何时耗尽，菲尔知道自己是否在享用早餐。在所有这些示例中，代理可以使用其经验改进其性能。国际象棋棋手改进他用来评估位置的直觉，从而改进他的比赛; 小羚羊提高了奔跑的效率; 菲尔学习流线型地做早餐。在开始时，代理带给任务的知识——无论是以前的相关任务经验，还是通过设计或演进构建的相关任务——会影响什么是有用的，什么是容易学习的，但是与环境的交互对于调整行为以利用任务的特定特性是必不可少的。

除了代理和环境之外，还可以识别增强学习系统的四个主要子元素: 策略、奖励信号、值函数，以及可选的环境模型。策略定义了学习代理在给定时间的行为方式。粗略地说，政策是一种从感知的环境状态映射到在这些状态下要采取的行动的映射。它对应于心理学上所谓的一套刺激反应规则或联想。在某些情况下，策略可能是一个简单的函数或查找表，而在其他情况下，它可能涉及大量的计算，比如搜索过程。策略是增强学习代理的核心，因为它本身就足以决定行为。一般来说，策略可能是随机的，为每个动作指定概率。奖励信号定义了强化学习问题的目标。在每个时间步骤上，环境向增强学习代理发送一个称为奖励的单个数字。代理人的唯一目标是在长期内获得最大的回报。因此，奖励信号定义了对代理来说什么是好事，什么是坏事。在生物系统中，我们可能认为奖励类似于快乐或痛苦的体验。它们是代理所面临的问题的直接和明确的特征。奖励信号是改变政策的主要依据; 如果一个由策略选择的动作后面跟着低回报，那么该策略可能会被更改，以便在将来选择该情况下的其他动作。一般来说，奖励信号可能是环境状态和所采取行动的随机函数。而奖励信号在即时意义上表示什么是好的，而值函数指定了什么是长期的好。粗略地说，一个国家的价值是一个代理人在未来累积的回报总额，从这个国家开始。虽然奖励决定了环境状态的直接的、内在的可取性，但价值表明在考虑到可能遵循的国家和这些国家的可获得的奖励之后，国家的长期可取性。例如，一个国家可能总是会得到一个低的即时回报，但仍然有很高的价值，因为它经常被其他的国家所效仿，获得高回报。反之亦然。打个人类的比方，奖励有点像快乐 (如果高的话) 和痛苦 (如果低的话)，而价值则对应于对我们的环境处于特定状态时的高兴或不愉快程度的更精确和更有远见的判断。在某种意义上，奖励是首要的，而价值作为对奖励的预测是次要的。没有奖励就没有价值，估计价值的唯一目的就是获得更多的奖励。然而，我们最关心的是我们在制定和评估决策时最关心的价值观。行动选择是基于价值判断。我们寻求能带来最高价值的行为，而不是最高回报的行为，因为这些行为在长期内为我们获得最大的回报。不幸的是，要确定价值比确定回报要难得多。奖励基本上是由环境直接给出的，但是值必须通过一个代理在其整个生命周期中所做的观察序列来估计和重新估计。事实上，我们考虑的所有强化学习算法中最重要的部分是 a

1.4 限制和范围

有效估计值的方法。价值评估的核心作用可以说是在过去 60 年中学到的关于强化学习的最重要的东西。一些强化学习系统的第四个也是最后一个要素是环境模型。这是一种模仿环境行为的东西，或者更一般地说，它允许对环境的行为进行推断。例如，给定一个状态和动作，模型可能预测下一个状态和下一个奖励的结果。模型用于规划，我们指的是在实际经历之前，通过考虑可能的未来情况来决定行动的任何方式。解决使用模型和计划的强化学习问题的方法被称为基于模型的方法，而不是简单的无模型的方法，后者是显式的试错学习者——被视为几乎与计划相反。在第 8 章，我们探索强化学习系统，通过尝试和错误同时学习，学习环境模型，并使用模型进行规划。现代强化学习的范围从低水平的、反复试验的学习到高层次的、深思熟虑的计划。

强化学习在很大程度上依赖于状态作为策略和值函数的输入的概念，并且作为模型的输入和输出。非正式地，我们可以将状态视为在特定时间向代理传递“环境如何”的某种感觉的信号。我们在这里使用的状态的正式定义是由马尔可夫决策过程框架在第 3 章中给出的。然而，更一般地说，我们鼓励读者遵循非正式的含义，并将国家视为代理可以获得的关于其环境的任何信息。实际上，我们假设状态信号是由某些预处理系统产生的，而这些预处理系统名义上是代理环境的一部分。在本书中，我们不讨论构造、更改或学习状态信号的问题 (除了在第 17.3 节中简要介绍)。我们采取这种做法不是因为我们认为国家代表性不重要，而是为了充分集中于决策问题。换句话说，我们在这本书中关注的不是设计状态信号，而是决定以任何状态信号可用的函数来采取什么动作。我们在本书中考虑的大多数强化学习方法都是围绕估计价值函数来构建的，但并不是一定要这样做来解决强化学习的问题。例如，求解方法

如遗传算法、遗传规划、模拟退火和其他优化方法都不估计值函数。这些方法应用多个静态策略，每个策略在一段时间内以独立的环境实例进行交互。获得最多奖励的策略，以及随机变化的策略，将被传递到下一代策略中，并重复这个过程。我们称这些进化方法为进化方法，因为它们的运作类似于生物进化产生具有熟练行为的生物体的方式，即使它们在它们的一生中没有学习过。如果政策的空间足够小，或者可以构造成好的政策。

常见的或容易找到的——或者如果大量的时间用于搜索，那么进化的方法是有效的。此外，进化方法在学习代理无法感知其环境完整状态的问题上具有优势。我们的重点是强化学习方法，这些方法在与环境交互时学习，而进化方法则不这样做。在许多情况下，能够利用个体行为交互细节的方法比进化方法更有效。进化方法忽略了强化学习问题的许多有用结构: 它们没有利用这样一个事实，即它们正在寻找的策略是一个从状态到行为的函数; 他们没有注意到一个个体在其一生中通过了哪些状态，或者它选择了哪些行为。在某些情况下，这些信息可能具有误导性 (例如，当状态被误解时)，但更经常的是，它应该使搜索更有效。虽然进化和学习有许多共同的特点，并且自然地共同工作，我们不认为进化方法本身特别适合强化学习问题，因此，我们在本书中不讨论它们。

1.5 一个扩展的例子: 井字游戏

为了说明强化学习的一般概念，并将其与其他方法进行对比，我们接下来将更详细地考虑单个示例。

想想熟悉的孩子玩的井字游戏。两个玩家轮流在一块三乘三的棋盘上玩。一个玩家玩 Xs 游戏，另一个玩家玩 Os 游戏，直到其中一个玩家在游戏中以一排三分的成绩获胜，水平、垂直或对角线排列，就像 X 玩家在游戏中向右显示的那样。如果棋盘上没有一个玩家连续得到 3 个，那么游戏就是平局。因为一个有技巧的球员可以打得永远不会输，所以让我们假设我们在和一个不完美的球员比赛，这个球员的表现有时是错误的，让我们可以赢。目前, 事实上，让我们考虑平局和损失对我们同样不利。我们该如何构建一个能够发现对手在比赛中不完美之处，并学会最大限度地增加获胜机会的球员呢? 虽然这是一个简单的问题，但不能通过经典的技术以令人满意的方式解决它。例如，博弈论中的经典“极小极大”解在这里是不正确的，因为它假定对手有一种特定的玩法。例如，一个小游戏玩家永远不会达到它可能失败的游戏状态，即使事实上它总是从那个状态中获胜，因为对手的不正确的游戏。经典的顺序决策问题的优化方法，如动态规划，可以计算任何对手的最优解，但需要输入一个完整的对手的规格，包括对手在每个板状态下的每个动作的概率。让我们假定，这个信息不是为这个问题而事先获得的，因为它不是为大多数实际感兴趣的问题而提供的。另一方面，这种信息可以通过经验来估计，在这种情况下，可以通过与对手进行多场比赛来估计。这是最好的办法

在这个问题上，首先要学习一个对手行为的模型，直到一定程度的自信，然后应用动态规划来计算一个给定近似对手模型的最优解。最后，这与我们稍后在本书中研究的一些强化学习方法并没有什么不同。一种应用于这个问题的进化方法将直接搜索可能的策略的空间，从而有可能赢得对手的胜利。在这里，策略是一条规则，告诉玩家在游戏的每一种状态下该做什么——三乘三的棋盘上所有可能的 Xs 和 Os 配置。对于每一种策略，通过对对手进行一定数量的游戏来估计其获胜的可能性。然后，该评估将指导下一步考虑哪些政策或政策。一种典型的进化方法是在政策空间中爬升，不断地生成和评估政策，试图获得渐进的改进。或者，也许可以使用一种遗传类型的算法来维护和评估一组策略。实际上可以应用数百种不同的优化方法。下面是利用值函数的方法来解决井字问题的方法。首先，我们要建立一个数字表，每个数字对应游戏的可能状态。每个数字都是我们从那个州获胜的概率的最新估计值。我们把这个估计看作是状态的值，整个表是学习的值函数。状态值高于国家 B, 或被认为是比状态 B “更好”, 如果当前的概率的估计我们赢得来自一个高于 B . 假设我们总是玩 x, 然后连续所有州有三个 Xs 获胜的概率是 1, 因为我们已经赢了。同样，对于所有连续有三个 Os 的状态，或者被填满的状态，正确的概率是 0，因为我们无法从它们中获胜。我们将所有其他州的初始

值设为 0.5，表示我们有 50 然后我们和对手进行了很多比赛。为了选择我们的移动，我们检查每一个可能的移动的状态 (黑板上每个空格对应一个)，并在表中查找它们的当前值。大多数时候，我们都是贪婪地前进，选择的移动会导致价值最高的状态，也就是说，赢得的概率最高。然而，我们偶尔会从其他动作中随机选择。这些被称为探索性行动，因为它们使我们经历了我们可能从未见过的状态。在游戏中进行和考虑的一系列动作可以如图 1.1 所示。当我们在玩的时候，我们改变了我们在游戏中发现自己的状态的价值。我们试图使他们更准确地估计获胜的可能性。为此，我们“备份”了每个贪婪移动到状态之前的状态的值，如图 1.1 中的箭头所示。更准确地说，前面状态的当前值被更新为更接近后面状态的值。这可以通过将较早状态的值移动到较晚状态的值的一小部分来实现。如果我们让 St 表示贪婪移动前的状态，St+1 表示移动后的状态，则更新 St 的估计值为 V (St)

V(St)←(St)+

V(圣 + 1)−V(St)

起始位置开始的位置

图 1.1: 一组井字棋的动作。实心的黑线代表在游戏中所采取的动作; 虚线代表我们 (我们的强化学习玩家) 考虑过但没有做出的动作。我们的第二个行动是一个探索性的举动, 这意味着它是即使另一个兄弟姐妹, 一个导致 e∗, 排名更高。探索性移动不会导致任何学习，但是我们其他的每一个移动都会导致更新，如红色箭头所示，其中估计的值将从后面的节点移动到前面的节点，如文中所述。

在 是一个小的积极的部分称为步长参数, 影响学习的速度。这个更新规则是 temporal-difference 学习方法的一个例子, 所谓的, 因为它的变化是基于不同,V(圣 + 1)−V(St), 估计之间的连续两次。上面描述的方法在这个任务上执行得很好。例如，如果步长参数随时间适当减小，那么对于任何固定的对手，该方法收敛到我们的玩家在最佳发挥下从每个状态中获胜的真实概率。此外，所采取的行动 (除了探索性的行动) 实际上是针对这个 (不完美的) 对手的最佳行动。换句话说，该方法收敛于与该对手博弈的最优策略。如果步长参数没有随着时间的推移一直降低到零，那么这个玩家也可以很好地对抗那些慢慢改变他们的游戏方式的对手。这个例子说明了进化方法和学习价值函数的方法之间的区别。为了评估一个策略，演化方法将策略固定，并对对手进行许多游戏，或者使用对手的模型来模拟许多游戏。获胜的频率给出了概率的无偏估计

用该策略获胜，并可用于指导下一个策略的选择。但是每一个政策的改变都是在许多游戏之后才进行的，并且只有每个游戏的最终结果被使用: 在游戏期间发生的事情被忽略了。例如，如果玩家赢了，那么他在游戏中的所有行为都得到了认可，这与具体的动作对获胜的关键程度无关。甚至对从未发生过的动作给予赞扬! 与此相反，值函数方法允许对各个状态进行评估。最后，进化函数和价值函数方法都是搜索策略的空间，但是学习一个价值函数利用了在过程中可用的信息。这个简单的例子说明了强化学习方法的一些关键特性。首先，强调的是在与环境交互时的学习，在这种情况下，与对手进行交互。第二，有一个明确的目标，正确的行为需要考虑到自己选择的延迟效应的计划或远见。例如，简单的增强学习玩家将学会设置多移动陷阱，为一个目光短浅的对手。增强学习解决方案的一个显著特点是，它不需要使用对手的模型，也不需要对未来状态和动作的可能序列进行显式搜索，就可以实现规划和预测的效果。虽然这个例子说明了强化学习的一些关键特性，但是它是如此简单，以至于它可能会给人这样的印象，强化学习比实际的要有限。虽然井字游戏是一种双人游戏，强化学习也适用于没有外部对手的情况，也就是说，在“对抗自然”的情况下。强化学习也不局限于行为分解成不同阶段的问题，比如“井字游戏”的独立游戏，只有在每一集结束时才会有奖励。当行为无限期地持续下去，以及任何时候都能收到不同程度的奖励时，它同样适用。强化学习也适用于那些甚至不像井字游戏那样分解成离散时间步骤的问题。一般原则也适用于持续时间的问题，尽管理论变得更加复杂，我们在这个介绍的处理中省略了它。井字棋有一个相对较小的有限状态集，当状态集非常大甚至无穷大时，可以使用强化学习。例如，Gerry Tesauro(1992, 1995) 将上面描述的算法与一个人工神经网络相结合，以学习玩具有大约 1020 个状态的西洋双陆棋。有了这么多的状态，你不可能体验到其中的一小部分。Tesauro 的程序

学得比以前的程序好多了，最终也比世界上最好的人类玩家强 (第 16.1 节)。人工神经网络为程序提供了从经验中归纳的能力，以便在新的状态中，它根据以前面对的相似状态所保存的信息来选择动作，这是由它的网络决定的。强化学习系统在如此大的状态集中如何工作与它从过去的经验中如何恰当地归纳密切相关。正是在这个角色中，我们最需要的是强化学习的监督学习方法。人工神经网络和深度学习 (第 9.6 节) 不是唯一的，也不一定是最好的方法。在这个井字游戏的例子中，学习是在没有知识的情况下开始的

游戏规则，但强化学习绝不意味着学习和智力的一张白板。相反，先前的信息可以以各种对有效学习至关重要的方式纳入强化学习 (例如，参见 9.5、17.4 和 13.1 节)。我们也可以在井字游戏中获得真实的状态，而强化学习也可以在部分状态被隐藏，或者当不同的状态在学习者看来是相同的时候被应用。最后，井字游戏的玩家能够向前看，知道每一个可能的移动会产生什么状态。要做到这一点，它必须有一个游戏模型，让它能够预见到它的环境将如何随着它可能永远不会做出的动作而变化。许多问题都是这样，但在其他问题上，甚至缺乏行动效果的短期模型。强化学习可以应用于任何一种情况。模型不是必需的，但是如果模型是可用的或可以学习的，那么模型可以很容易地使用 (第 8 章)。另一方面，强化学习方法根本不需要任何环境模型。没有模型的系统甚至不能考虑它们的环境如何随着单个动作而变化。从这个意义上说，井字游戏玩家对其对手是没有模型的: 它没有任何对手的模型。由于模型必须具有相当的准确性，因此，当解决问题的真正瓶颈是构建一个足够精确的环境模型的困难时，无模型方法比更复杂的方法具有优势。无模型方法也是基于模型的方法的重要构建块。在本书中，在讨论如何将无模型方法用作更复杂的基于模型的方法的组件之前，我们将用几章讨论无模型方法。强化学习可以在系统的高水平和低水平上使用。虽然井字游戏玩家只学习了游戏的基本动作，但没有什么能阻止强化学习在更高的层次上进行，在更高的层次上，每一个“动作”本身都可能是一种精心设计的解决问题的方法的应用。在分层学习系统中，强化学习可以同时在几个层次上进行。练习 1.1: 自玩假设，上面描述的强化学习算法不是与随机对手对抗，而是与自己对抗，双方都在学习。你认为在这种情况下会发生什么? 它会学到不同的政策吗选择移动吗? ? 对称性: 由于对称性，很多井字的位置看起来不同，但实际上是相同的。我们如何修改上面描述的学习过程来利用这一点呢? 这种改变将在哪些方面改善学习过程? 现在再想想。假设对手没有利用对称性。既然如此，我们应该这样做吗? 那么，对称等价的位置应该是正确的吗一定有相同的价值吗? ? 练习 1.3: 贪心游戏假设强化学习玩家是贪心的，也就是说，它一直在做使它达到它认为最好的位置的动作。它会比一个不贪心的玩家玩得更好，还是更糟? 会发生什么问题?? 练习 1.4: 从探索中学习假设学习更新发生在所有动作之后，包括探索动作。如果步长参数适当减少

随着时间的推移 (但不是探究的趋势)，状态值会收敛到一组不同的概率。当我们这样做的时候 (从概念上)，当我们不这样做的时候，这两组概率是什么? 假设我们继续进行探索性的行动，那一组概率可能会更好。学习吗? 哪个会导致更多的胜利? ? 练习 1.5: 你能想出其他方法来提高强化学习玩家吗? 你能想出更好的办法来解决井字游戏的问题吗作为构成? ?

1.6 总结

强化学习是理解和自动化目标导向学习和决策的一种计算方法。它与其他计算方法的区别在于，它强调由代理进行学习，而不是直接与环境交互，而不需要模范监督或完整的环境模型。在我们看来，强化学习是第一个认真处理从与环境交互中产生的计算问题，以实现长期目标的领域。强化学习使用马尔可夫决策过程的正式框架，以状态、行为和奖励的形式定义学习主体与其环境之间的交互。这个框架旨在成为表示人工智能问题的基本特征的一种简单方法。这些特征包括因果感、不确定性和不确定性，以及明确目标的存在。价值和价值函数的概念是我们在本书中考虑的大多数强化学习方法的关键。我们的立场是，在政策空间中，价值函数对于有效搜索非常重要。价值函数的使用区分了强化学习方法和进化方法，后者直接在整个政策评估的指导下在政策空间中搜索。

1.7 强化学习的早期历史

强化学习的早期历史有两个主要的主线，分别是长期和丰富的，在现代强化学习中相互交织。有一条线是关于尝试和错误的学习，它起源于动物学习心理学。这一思路贯穿于人工智能的早期研究，并在 20 世纪 80 年代早期引领了强化学习的复兴。第二个线程是利用值函数和动态规划的最优控制问题及其解决方案。在大多数情况下，这条线并不涉及学习。这两个线程大部分是独立的，但在某种程度上与第三个不同的线程相互关联，在这一章中，在井字的例子中使用的时间差方法就不那么明显了。这三个线索在 20 世纪 80 年代末汇聚在一起，产生了我们在本书中展示的现代强化学习领域。

关注试错学习的线索是我们最熟悉的，也是我们在这段简短的历史中最有发言权的。然而，在此之前，我们简要地讨论了最优控制线程。“最优控制”一词在 20 世纪 50 年代末开始使用，用来描述设计一个控制器来最小化或最大化一个动态系统随时间变化的行为度量的问题。解决这个问题的方法之一是理查德 • 贝尔曼和其他人在 20 世纪 50 年代中期提出的通过扩展 19 世纪的汉密尔顿和雅可比理论。这种方法使用动态系统的状态和值函数的概念，或者“最优返回函数”来定义一个函数方程，现在通常称为 Bellman 方程。通过求解这个方程来求解最优控制问题的方法被称为动态规划 (Bellman, 1957a)。Bellman (1957b) 还引入了最优控制问题 Markov decision process (MDPs) 的离散随机版本。Ronald Howard(1960) 设计了 MDPs 的策略迭代方法。所有这些都是现代强化学习理论和算法的基础要素。动态规划被广泛认为是解决一般随机最优控制问题的唯一可行方法。它受 Bellman 所称的“维度的诅咒”的影响，这意味着它的计算需求随状态变量的数量呈指数增长，但它仍然比其他一般方法更有效和更广泛地适用。自 20 世纪 50 年代末以来，动态编程得到了广泛的发展，包括对部分可观测的 MDPs 的扩展 (由洛夫乔伊调查，1991)，许多应用 (由怀特调查，1985,1988,1993)，近似方法 (由 Rust 调查，1996) 和异步方法 (Bertsekas, 1982, 1983)。许多优秀的动态规划的现代处理方法是可用的 (例如，Bertsekas, 2005, 2012;Puterman,1994; 罗斯,1983;1982 年惠特尔,1983)。Bryson(1996) 提供了最优控制的权威历史。一方面，最优控制与动态规划之间的联系，另一方面，学习之间的联系很难被识别。我们不能确定这种分离的原因是什么，但是它的主要原因可能是所涉及的学科和它们的不同目标之间的分离。动态规划作为一种离线计算的普遍观点，在本质上依赖于精确的系统模型和 Bellman 方程的解析解的基础上，也可能有所贡献。此外，最简单的动态编程形式是一种向后进行的计算，这使我们很难看到它是如何参与到一个必须朝前进行的学习过程中来的。一些早期的动态编程工作，如 Bellman 和 Dreyfus(1959) 的工作，现在可以归类为遵循学习方法。Witten(1977) 的作品 (下文讨论) 肯定是学习和动态规划思想的结合。Werbos(1987) 明确主张动态规划和学习方法的更大的相互关系，以及动态规划与理解神经和认知机制的相关性。对于我们来说，直到 1989 年克里斯 • 沃特金斯 (Chris Watkins) 的作品，在使用 MDP 形式主义的强化学习的方法被广泛采用的情况下，才出现了动态规划方法与在线学习的完全融合。从那时起，这些关系被许多研究者广泛发展，尤其是迪米特里 • 贝尔采卡斯

John tsiklis(1996) 发明了“神经动态规划”一词，指的是动态规划和人工神经网络的结合。目前使用的另一个术语是“近似动态规划”。这些不同的方法强调了这门学科的不同方面，但它们都与强化学习一样，都对绕过动态规划的经典缺陷感兴趣。我们认为最优控制下的所有工作，在某种意义上，也是强化学习中的工作。我们将强化学习方法定义为解决强化学习问题的有效方法，现在很明显，这些问题与最优控制问题密切相关，特别是像 MDPs 这样的随机最优控制问题。因此，我们必须考虑最优控制的解决方法，如动态规划，也要考虑强化学习方法。因为几乎所有的常规方法都需要完全了解系统才能被控制，所以说它们是强化学习的一部分有点不自然。另一方面，许多动态规划算法是递增和迭代的。像学习方法一样，他们通过不断的逼近逐渐得到正确的答案。正如我们在本书其余部分所展示的，这些相似之处远不止表面现象。完全知识和不完全知识的情况下的理论和解决方法是如此紧密地联系在一起，我们认为它们必须作为同一主题的一部分来考虑。现在让我们回到通向现代强化学习领域的另一条主线，这条主线围绕着试错学习的思想。我们只讨论这里的主要联系人，在第 14.3 节

中更详细地讨论这个主题。根据美国心理学家 r s Woodworth(1938) 试误学习的想法可以追溯到 1850 年代, 亚历山大 • 贝恩的讨论学习的“摸索和实验”, 更明确地向英国动物行为学家和心理学家康威劳埃德 • 摩根的 1894 使用的术语来描述他的观察动物的行为。也许第一个简洁地表达了试错学习的本质的原则是爱德华桑代克:

在对相同情况作出的几种反应中，在其他条件相同的情况下，伴随或紧随其后的是对动物的满足，与这种情况更紧密地联系在一起，因此，当它再次出现时，它们将更有可能再次出现; 在其他条件相同的情况下，那些伴随或紧随其后的动物会削弱它们与这种情况的联系，这样，当这种情况再次发生时，它们就不太可能发生。满足感或不舒服程度越高，这种联系就越牢固或减弱。(桑代克,1911,p . 244) 桑代克把这称为“效应定律”，因为它描述了强化事件对选择行为倾向的影响。后来，桑代克修改了法律，以便更好地解释关于动物学习的后续数据 (比如奖励和惩罚效果之间的差异)，而法律的各种形式在学习理论家中引起了相当大的争议 (例如，参见 Gallistel, 2005; 伯恩斯坦,1970; 金布尔、1961、1967;Mazur,1994)。尽管如此，作为一种或另一种形式的有效法则被广泛认为是许多行为背后的基本原则 (例如，Hilgard 和 Bower, 1975; 丹尼特,1978; 坎贝尔,1960;Cziko,1995)。它是有影响力的人的基础

克拉克 • 赫尔的学习理论 (1943,1952) 和斯金纳的影响实验方法 (1938)。在动物学习的语境中，“强化”一词是在桑代克表达“效应定律”之后才开始使用的，最早出现在这一语境中 (据我们所知)。巴甫洛夫将强化描述为一种行为模式的强化，这种行为模式是由于动物受到刺激——一种强化——在与另一种刺激或反应的适当时间关系中。一些心理学家将强化的概念扩展到包括弱化和强化行为，并扩展了强化的概念，包括可能省略或终止刺激。要被认为是强化物，强化物被收回后，强化物或弱化物必须继续存在; 仅仅吸引动物注意力或激励动物行为而不产生持久变化的刺激物不会被认为是强化物。在计算机中实现试错学习的想法最早出现在关于人工智能的可能性的思想中。在 1948 年的一份报告中，艾伦 • 图灵 (Alan Turing) 描述了一种“愉悦-痛苦系统”的设计，该系统遵循的是效果定律:

当到达一个未确定操作的配置时，将对丢失的数据进行随机选择，并在描述中进行适当的输入，并将其应用于描述中。当疼痛刺激发生时，所有试探性的条目都被取消，当快乐刺激发生时，它们都是永久性的。(图灵,1948) 许多精巧的机电设备被制造出来，演示了反复试验的学习。最早的机器可能是由托马斯 • 罗斯 (1933 年) 建造的机器，它能够通过一个简单的迷宫找到路，并通过开关的设置记住路径。在 1951 W。格雷沃特建造了他的“机械乌龟”(沃尔特，1950 年) 的一个版本，他有一种简单的学习方式。1952 年，克劳德 • 香农 (Claude Shannon) 演示了一种名为忒修斯 (Theseus) 的老鼠，它通过反复试验找到迷宫的路径，迷宫本身通过磁铁和地底继电器记住成功的方向 (参见香农 (Shannon, 1951))。J. a . Deutsch(1954) 基于他的行为理论 (Deutsch, 1953) 描述了一台求解迷宫的机器，它与基于模型的强化学习有一些共同的特性 (第八章)。马文 • 明斯基 (1954) 讨论了强化学习的计算模型，并描述了他构建的一个模拟机器，由他称之为“随机神经-模拟强化计算器”的组件组成，这些组件旨在模拟大脑中可修改的突触连接 (第 15 章)。建造机电学习机器让位于编程数字计算机来执行各种类型的学习，其中一些实现了试错学习。Farley 和 Clark(1954) 描述了一种神经网络学习机器的数字模拟，这种机器通过反复试验来学习。但他们的兴趣很快从试错学习转移到泛化和模式识别，即从强化学习转移到监督学习 (Clark and Farley, 1955)。这一模式开始

对这些类型的学习之间的关系感到困惑。许多研究人员似乎认为，他们在研究强化学习时，实际上是在研究监督学习。例如，像 Rosenblatt(1962) 和 Widrow 和 Hoff(1960) 这样的人工神经网络先驱显然是受到强化学习的激励——他们使用奖励和惩罚的语言——但他们研究的系统是监督学习系统，适合于模式识别和知觉学习。即使在今天，一些研究人员和教科书也淡化或模糊了这些类型的学习之间的区别。例如，一些人工神经网络教科书使用“试错法”一词来描述从训练例子中学到的网络。这是一种可以理解的混淆，因为这些网络使用错误信息来更新连接权值，但这忽略了试错学习的本质特征，即在不依赖于正确操作的知识的评估反馈的基础上选择操作。部分由于这些混淆，真正的试错学习的研究在 20 世纪 60 年代和 70 年代变得罕见，尽管有明显的例外。20 世纪 60 年代，在工程文献中首次使用“强化”和“强化学习”这两个术语来描述试错学习的工程应用 (例如，华尔兹和傅，1965; 孟德尔,1966; 傅,1970;

孟德尔和麦克拉伦,1970)。特别有影响力是明斯基的论文“步骤人工智能”(明斯基,1961), 相关的试错学习讨论几个问题, 包括预测, 期望, 和他所谓的基本 credit-assignment 复杂的强化学习系统的问题: 你如何分配信贷成功的决策, 可能是参与生产吗? 我们在这本书中讨论的所有方法，在某种意义上，都是为了解决这个问题。明斯基的论文今天很值得一读。在接下来的几段中，我们讨论了在 20 世纪 60 和 70 年代对真正的试错学习的计算和理论研究相对忽视的一些例外和部分例外情况。新西兰研究人员约翰 • 安德烈亚 (John Andreae) 的工作是一个例外。他开发了一种名为 STeLLA 的系统，通过与周围环境的反复试验来学习。这个系统包括了一个世界的内部模型，以及后来处理隐藏状态问题的“内部独白”(Andreae, 1963, 1969a,b)。Andreae 后来的作品 (1977) 更强调从老师那里学习，但仍然包括通过尝试和错误来学习，创造新事件是系统的目标之一。这项工作的一个特点是“泄漏过程”，在 Andreae(1998) 中更详细地阐述了这个过程，它实现了一种信贷分配机制，类似于我们所描述的备份更新操作。不幸的是，他的开创性研究并不为人所知，对后续的强化学习研究影响不大。最近的总结有 (Andreae, 2017a,b)。更有影响力的是 Donald Michie 的作品。在 1961 年和 1963 年，他描述了一个简单的试错学习系统，用来学习如何玩“三趾棋”(或“鹦鹉”和“十字”)。它由一个火柴盒组成，每个火柴盒包含了一些颜色的珠子，每个可能的移动都有不同的颜色。通过

从火柴盒中随机绘制出与当前游戏位置相对应的珠子，可以确定威胁的移动。当游戏结束时,在游戏中用来奖励或惩罚威胁的决定的盒子里添加或移除珠子。Michie 和 Chambers(1968) 描述了另一种被称为“欢乐合唱团”(Game Learning Expectimaxing Engine) 的井字游戏强化学习器，以及名为 box 的强化学习控制器。他们用箱子来学习平衡一根杆子和一辆可移动的车，因为只有当杆子落下或车走到轨道末端时才会出现故障信号。这项任务改编自 Widrow 和 Smith(1964) 的早期工作，他们使用了监督学习方法，假设老师已经能够平衡杆端。麦奇和钱伯斯版本的杆平衡是在不完全知识条件下强化学习任务的最佳早期范例之一。它影响了后来的强化学习工作，从我们自己的一些研究开始 (Barto, Sutton, and Anderson, 1983; 萨顿,1984)。Michie 始终强调试验、错误和学习作为人工智能的基本方面的作用 (Michie, 1974)。Widrow, Gupta，和 Maitra(1973) 修改了 Widrow 和 Hoff(1960) 的最小均方 (LMS) 算法，以产生一个增强学习规则，可以从成功和失败的信号中学习，而不是从训练示例中学习。他们将这种学习方式称为“选择性自适应”，并将其描述为“与批评家一起学习”，而不是“与老师一起学习”。他们分析了这个规则，并展示了它是如何学会玩 21 点的。这是 Widrow 单独对强化学习的尝试，他对监督学习的贡献要大得多。我们对“批评家”一词的使用源自 Widrow、Gupta 和 Maitra 的论文。布坎南 (Buchanan)、米切尔 (Mitchell)、史密斯 (Smith) 和约翰逊 (Johnson)(1978) 在机器学习的背景下独立地使用了“批评家”一词 (参见 Dietterich 和布坎南 (Buchanan)，1984)，但对他们来说，批评家是一个能够做的不仅仅是评价性能的专家系统。学习自动机的研究对试错线有更直接的影响，从而导致了现代强化学习的研究。这些方法是用来解决一个非联想的，纯粹的选择学习的问题，被称为 k-武装土匪，类似于老虎机，或“独臂赌博机”，除了 k 个杠杆 (见第二章)。学习自动机是一种简单的，低内存的机器，用于提高这些问题中奖励的可能性。学习自动机起源于 20 世纪 60 年代的俄国数学家和物理学家 m.l. Tsetlin 和他的同事们 (在 1973 年的 Tsetlin 出版后出版)，从那时起就在工程领域内得到了广泛的发展 (见 Narendra 和 Thathachar, 1974, 1989)。这些发展包括随机学习自动机的研究，这是一种基于奖励信号更新行动概率的方法。Harth 和 Tzanakou(1974) 的 Alopex 算法 (用于模式提取算法) 虽然没有在随机学习自动机的传统中发展起来，但它是一种随机的方法，用于检测行为和强化之间的相关性，影响了我们早期的一些研究 (Barto, Sutton，和 Brouwer, 1981)。随机学习自动机是心理学早期工作的先兆，从 William Estes(1950) 研究学习的统计理论开始，并由其他人进一步发展 (例如 Bush 和 Mosteller, 1955; 斯特恩伯格,1963)。心理学中发展起来的统计学习理论被研究者采用

经济学，导致了这一领域的研究，致力于强化学习。这项工作始于 1973 年，当时布什和大多数人的学习理论被应用于古典经济模型的收集 (Cross, 1973)。这项研究的一个目标是研究比传统理想化的经济主体更像真人的人工制剂 (Arthur, 1991)。该方法扩展到博弈论背景下的强化学习研究。经济学中的强化学习在很大程度上独立于人工智能中强化学习的早期工作，

尽管博弈论仍然是这两个领域的研究热点 (超出了本书的范围)。卡默勒 (2011) 讨论了强化学习的传统经济学和 Nowé,Vrancx,De Hauwere(2012) 提供了一个概述的主题从的角度多主体扩展到这本书中, 我们介绍的方法。在游戏理论的背景下，强化是一个与强化学习截然不同的学科，用于玩井字游戏、跳棋和其他娱乐游戏。例如，参见 Szita(2012) 对增强学习和游戏这一方面的概述。John Holland(1975) 概述了基于选择原则的适应性系统的一般理论。他早期的工作主要涉及试验和错误，主要是非联想形式，如进化方法和 k 武装匪徒。1976 年和 1986 年，他更全面地介绍了分类器系统，真正的强化学习系统，包括关联和价值函数。Holland 分类器系统的一个关键组成部分是信用分配的“桶-旅算法”，它与我们的井-塔-足趾示例中使用的时间差异算法密切相关，并在第 6 章中进行了讨论。另一个关键组成部分是遗传算法，这是一种进化方法，其作用是进化出有用的表示法。分类器系统已经被许多研究者广泛发展形成的一个主要分支强化学习研究 (由 Urbanowicz 和摩尔,2009), 但遗传算法不考虑强化学习系统的自己也得到了更多的关注, 因为有其他的进化计算方法 (例如, 福格尔, 欧文斯和沃尔什 1966 年 Koza,1992)。哈利 • 克洛普夫 (1972 年，1975 年，1982 年) 是人工智能强化学习的最主要负责人。Klopf 认识到，随着学习研究者几乎完全专注于监督学习，适应性行为的基本方面正在丧失。Klopf 认为，缺少的是行为的享乐性方面，即从环境中获得某种结果的动力，即控制环境达到预期目的，远离不期望的目的 (见第 15.9 节)。这是反复试验学习的基本思想。Klopf 的观点对作者的影响尤其大，因为我们对它们的评估 (Barto 和 Sutton, 1981a) 导致我们对监督学习和强化学习之间的区别的理解，以及我们最终对强化学习的关注。我们和同事完成的许多早期工作是为了表明强化学习和监督学习确实是不同的 (Barto, Sutton，和 Brouwer, 1981;Barto 和萨顿,1981 b;Barto 和阿南丹,1985)。其他研究显示强化学习如何解决人工神经网络学习中的重要问题，特别是如何为多层网络生成学习算法 (Barto, Anderson，和 Sutton, 1982;Barto 和安德森,1985;Barto、1985、1986;Barto 和约旦,1987)。

现在我们来看第三条关于强化学习的历史，关于时间差异学习。时间差学习方法的独特之处在于，它是由同一数量的时间连续估计的差异所驱动的，例如，在井字游戏中获胜的概率。这条线比另外两条线更小，也不那么明显，但它在这一领域中发挥了特别重要的作用，部分原因是时间差方法似乎是加强学习的新方法和独特之处。时间差异学习的起源部分是在动物学习心理学中，特别是在次级强化的概念中。次级强化物是一种刺激物与初级强化物 (如食物或疼痛) 相结合，因此，也具有类似的强化性质。Minsky(1954) 可能是第一个认识到这一心理学原理对人工学习系统很重要的人。Arthur Samuel(1959) 首先提出并实施了一种包括时间性差异的学习方法，作为他著名的跳棋项目的一部分 (第 16.2 节)。塞缪尔没有提及明斯基的作品，也没有提及与动物学习可能的联系。他的灵感显然来自克劳德 • 香农 (Claude Shannon)(1950) 的建议，即计算机可以被编程来使用一个评估函数来玩国际象棋，并且可以通过在线修改这个函数来改进它的游戏。(很可能香农的这些想法也影响了贝尔曼，但我们没有证据证明这一点。) 明斯基 (1961) 在他的“步骤”论文中广泛地讨论了塞缪尔的工作，提出了与次级强化理论的联系，包括自然强化和人工强化。正如我们所讨论的，在明斯基和萨缪尔的研究之后的十年里，很少有关于试错学习的计算工作，显然，在时间差异学习方面根本没有计算工作。1972 年，Klopf 将试错学习与时间性差异学习的一个重要组成部分结合起来。Klopf 感兴趣的是能够扩展到大型系统学习的原理，因此对局部强化的概念很感兴趣，即一个整体学习系统的子组件可以相互增强。他提出了“广义强化”的概念，即每个组成部分 (名义上是每个神经元) 都以强化的方式看待所有输入: 兴奋性输入作为奖励，抑制性输入作为惩罚。这与我们现在所知的时间差学习不一样，回想起来，它比塞缪尔的研究还远。另一方面，Klopf 将这一想法与试错学习联系起来，并将其与庞大的动物学习心理学实证数据库联系起来。Sutton (1978a,b,c) 进一步发展了 Klopf 的思想，特别是与动物学习理论的联系，描述了由时间连续的预测变化所驱动的学习规则。他和 Barto 完善了这些思想，并建立了基于时间差异学习的经典条件反射的心理学模型 (Sutton and Barto, 1981a;Barto 和萨顿,1982)。此外，还有其他几个影响深远的基于时间差异学习的经典条件作用心理学模型 (如 Klopf, 1988; 摩尔 et al .,1986; 萨顿和巴托，1987,1990)。当时发展起来的一些神经系统模型被很好地解释为时间差异学习 (Hawkins and Kandel, 1984; 伯恩 • 金里奇，巴克斯特，1990;Gelperin, Hopfield,

Tank, 1985;Tesauro,

1986;Friston et al.，1994)，虽然在大多数情况下没有历史联系。我们早期对时间差学习的研究深受动物学习理论和 Klopf 工作的影响。与明斯基的“步数”文件和塞缪尔的跳棋球员的关系直到后来才被承认。然而，到 1981 年，我们充分意识到前面提到的作为时间差和试错线程的一部分的所有工作。此时，我们开发了一种利用时间差学习与试错学习相结合的方法，称为表演-批判架构，并将此方法应用于 Michie 和 Chambers 的杆平衡问题 (Barto, Sutton, and Anderson, 1983)。该方法在 Sutton(1984) 的博士论文中得到了广泛的研究，并在 Anderson(1986) 的博士论文中扩展到利用反向传播神经网络。在此期间，Holland(1986) 以他的 bucket-brigade 算法的形式，在他的分类器系统中显式地引入了时间差概念。Sutton(1988) 将时变学习与控制分离，将其作为一种通用的预测方法，采取了关键步骤。这篇论文还介绍了 TD( ) 算法并证明了它的一些收敛性质。当我们在 1981 年完成我们关于演员-批评家架构的工作时，我们发现了 Ian Witten (1977, 1976a) 的一篇论文，这似乎是时间差异学习规则最早的出版。他提出了一种方法，我们现在称之为表格式 TD(0)，作为自适应控制器的一部分来解决 MDPs。这项研究于 1974 年首次提交给期刊出版，也出现在 Witten 1976 年的博士论文中。威滕的作品是安德烈亚早期对斯特拉和其他试错学习系统的研究的后代。因此，威滕 1977 年的论文跨越了强化学习研究的两条主线——试错学习和最优控制——同时对时间性差异学习做出了明显的早期贡献。1989 年，随着克里斯 • 沃特金斯 (Chris Watkins) 的 Q-learning 发展，时变差和最优控制线程得到了充分整合。这项工作扩展并整合了所有三个加强学习研究的线索。Paul Werbos(1987) 提出了自 1977 年以来反复试验学习和动态编程的融合，从而促成了这种融合。到沃特金斯的工作时，强化学习的研究有了巨大的增长，主要是在人工智能的机器学习子领域，也在更广泛的人工神经网络和人工智能领域。1992 年，Gerry Tesauro 的“西洋双陆棋”(backgammon) 游戏计划 TD-Gammon 取得了非凡的成功，这使该领域受到了更多的关注。在这本书的第一版出版后，神经科学发展了一个蓬勃发展的分支领域，专注于强化学习算法和神经系统强化学习之间的关系。许多研究人员指出，造成这种现象的最主要原因是时间差异算法的行为与多巴胺在大脑中产生神经元的活动之间有着不可思议的相似性 (Friston et al.，1994;Barto,1995;1995 年，Houk, Adams 和 Barto; 蒙太古，大安，塞杰诺斯基，1996; 还有舒尔茨、达安和蒙塔古，1997 年)。第 15 章对强化学习这一令人兴奋的方面进行了介绍。在最近的强化学习历史中所作的其他重要贡献太多了，在这个简短的叙述中无法提及; 我们在每一章的结尾都引用了其中的许多内容。

书目的言论

对于额外的强化学习的一般范围, 我们参考读者的书 Szepesvári(2010),Bertsekas 和 Tsitsiklis(1996),Kaelbling(1993), 和 Sugiyama Hachiya,Morimura(2013)。从控制或运营研究的角度来看，书中有 Si、Barto、Powell 和 Wunsch(2004)、Powell(2011)、Lewis and Liu(2012) 和 Bertsekas(2012)。曹教授的 (2009) 回顾了在其他学习和优化随机动态系统的方法中加强学习的方法。《机器学习》杂志的三期特别集中在强化学习上:Sutton(1992 年 2 月)、Kaelbling(1996 年) 和 Singh(2002 年)。Barto(1995 年 b) 提供了有用的调查; 凯尔布林，利特曼和摩尔 (1996);Keerthi 和 Ravindran(1997)。由 Weiring 和 van Otterlo(2012) 编辑的卷提供了对最近发展的一个很好的概述。

在这一章中，菲尔的早餐的例子是由 Agre(1988) 启发的。

1.5 建立了井字算例中所采用的时间差法第六章。

Part I

第一部分: 表列解方法

在本书的这一部分，我们用最简单的形式描述了强化学习算法的几乎所有核心思想: 状态和操作空间足够小，使得近似值函数可以表示为数组或表。在这种情况下，方法往往能找到精确的解，也就是说，它们往往能找到最优值函数和最优策略。这与本书下一部分所描述的近似方法不同，后者只找到近似解，但反过来可以有效地应用于更大的问题。本书的这一部分的第一章描述了关于强化学习问题的特殊情况的解决方法，在这种情况下只有一种状态，称为班迪特问题。第二章描述了我们在有限马尔可夫决策过程中处理的一般问题公式，以及它的主要思想，包括 Bellman 方程和 value 函数。接下来的三章描述了解决有限马尔可夫决策问题的三种基本方法: 动态规划、蒙特卡罗方法和时间差学习。每种方法都有其优点和缺点。动态规划方法在数学上得到了很好的发展，但是需要一个完整而准确的环境模型。蒙特卡罗方法不需要模型，概念上很简单，但是不适合逐步递增的计算。最后，时变差分方法不需要模型，而且是完全递增的，但是分析起来比较复杂。这些方法在效率和收敛速度方面也有不同。剩下的两章描述了如何将这三种方法组合起来以获得它们各自的最佳特性。在一章中，我们描述了蒙特卡罗方法的优点如何通过多步自举方法与时滞差分方法的优点相结合。在本书这一部分的最后一章中，我们展示了如何将时间差异学习方法与模型学习和规划方法 (如动态规划) 结合在一起，为表格式的强化学习问题提供一个完整统一的解决方案。

2. 第二章多臂赌博机

区别强化学习和其他类型学习的最重要的特征是，它使用训练信息来评估所采取的行动，而不是通过给出正确的行动来指导。这就产生了主动探索的需要，即对良好行为的明确搜索。纯粹的评价反馈表明所采取的行动有多好，而不是它是最好的还是最坏的行动。从另一方面来说，纯粹的指导性反馈表明了采取独立于实际采取的行动的正确行动。这种反馈是监督学习的基础，包括模式分类、人工神经网络和系统识别等大部分内容。从纯粹的形式来看，这两种反馈是截然不同的: 评价性反馈完全依赖于所采取的行动，而指导性反馈则独立于所采取的行动。在这一章中，我们研究了强化学习的评估方面，在一个简化的环境下，一个不涉及学习在一个以上的情况下行动的环境。这种非联想设置是大多数涉及评估反馈的先前工作已经完成的设置，它避免了完全强化学习问题的复杂性。通过研究这个案例，我们可以清楚地看到评估性反馈与指导性反馈的区别，但又可以与指导性反馈相结合。我们探索的特定的非联想、评估反馈问题是 k 武装匪徒问题的一个简单版本。我们用这个问题来介绍一些基本的学习方法，我们在后面的章节中扩展这些方法来应用于完整的强化学习问题。在这一章的结尾，我们通过讨论当班迪特问题成为联想时发生了什么，也就是说，当在多个情况下采取行动时，我们向完全强化学习问题迈进了一步。

2.1 韩国武装匪徒问题

考虑下面的学习问题。在 k 个不同的选项或操作中，你会反复面临选择。在每个选择之后，你会收到一个数字奖励，这个数字奖励来自一个固定的概率分布，这个分布取决于你选择的动作。你的

目标是在一段时间内最大化期望的总回报，例如，超过 1000 个动作选择，或时间步骤。这是 k-armed bandit 问题的最初形式，类似于老虎机 (老虎机) 或“one-armed bandit”，只不过它有 k 个杠杆，而不是一个杠杆。每一个动作选择就像是老虎机的一个杠杆的游戏，而回报就是中奖的回报。通过重复的动作选择，你要通过将你的动作集中在最好的杠杆上来最大化你的奖金。另一个类比是医生在一系列重病患者的实验治疗中做出选择。每个行动都是治疗的选择，每个奖励都是病人的生存或幸福。今天，“班迪特问题”这个术语有时被用来概括上面描述的问题，但是在这本书中，我们用它来指代这个简单的问题的情况。在我们的 k 武装匪徒问题中，每个 k 个动作都有预期的或平均的奖励，前提是这个动作是被选择的; 让我们把这叫做行动的价值。我们表示行动选择时间步 t 作为, 和相应的奖励的价值然后沿任意行动, 表示问 ∗(a), 是期望的奖励, 一个选择:

问 ∗(a)。= E(Rt | = 一个)。

如果您知道每个动作的值，那么解决 k -armed bandit 问题就很简单了: 您总是选择值最高的动作。我们假设您不确定地知道操作值，尽管您可能有估计。我们将时刻 a 的估计值 t 表示为 Qt(a)。我们希望 Qt(a) 接近问 ∗(a)。如果您保持对动作值的估计，那么在任何时候，至少有一个动作的估计值是最大的。我们称之为贪婪行为。当您选择其中一个动作时，我们说您正在利用您当前对操作的值的知识。如果您选择一个非贪婪操作，那么我们说您正在探索，因为这使您能够改进对非贪婪操作的值的估计。开发是一件正确的事情，可以使预期的回报最大化，但从长远来看，探索可能会产生更大的总体回报。例如，假设一个贪婪行为的值是确定的，而其他几个行为被估计为几乎一样好，但是有很大的不确定性。不确定的是，至少有一种行为可能比贪婪行为更好，但你不知道是哪一种。如果您有许多时间步骤可以进行操作选择，那么最好探索非贪婪的操作，并发现它们中哪个比贪婪的操作更好。短期而言，在探索过程中，回报较低，但长期而言回报较高，因为在你发现了更好的行为之后，你可以多次利用它们。由于不可能同时探索和利用任何单一的行动选择，人们经常提到探索和开发之间的“冲突”。在任何特定的情况下，探索或利用是否更好取决于估算的精确值、不确定性和剩余步骤的数量。关于 k -武装土匪的特殊数学公式和相关问题，有许多复杂的方法来平衡勘探和开发。

2.2 行为价值的方法

然而，这些方法中的大多数都对平稳性和先验知识做出了强有力的假设，这些假设在应用程序中或在我们在后续章节中考虑的完全强化学习问题中被违反或无法验证。当这些方法的假设不适用时，对这些方法的最佳性或有界损失的保证是没有什么安慰的。在这本书中，我们不担心探索和开发之间的复杂平衡; 我们只关心平衡它们。在这一章中，我们提出了几种简单的平衡方法来解决 k -armed bandit 问题，并证明它们比那些经常使用的方法要有效得多。平衡探索和开发的需要是强化学习中出现的一个独特的挑战; 我们版本的 k 武装匪徒问题的简洁性使我们能够以一种特别清晰的形式展示这一点。

我们首先更仔细地研究评估行为值的方法，以及使用评估做出行为选择决策的方法，我们统称为行为价值方法。回想一下，一个行为的真正价值是选择该行为时的平均回报。估计这一点的一种自然方法是将实际收到的奖励平均下来:

Qt(a)。= a 在 t 之前的奖励总和 a 在 t 之前的次数 = ? t−1 我 = 1 Ri • 艾 = ? t−1 我 = 1 Ai = (2.1)

其中谓词表示谓词为真时为 1，非为 0 的随机变量。如果分母为 0，那么我们将 Qt(a) 定义为某个默认值，比如 0。当分母趋于无穷时, 大数定律,Qt(a) 收敛于问 ∗(a)。我们称它为估算行动值的抽样平均方法，因为每个估计都是相关奖励样本的平均值。当然，这只是评估行动价值的一种方法，不一定是最好的。然而，现在让我们继续使用这个简单的估计方法，并转向如何使用估计来选择操作的问题。最简单的操作选择规则是选择一个估计值最高的操作，即上一节定义的贪婪操作之一。如果有一个以上的贪心行为，那么就会以任意的方式在其中进行选择，可能是随机的。我们把这个贪心行为选择方法写成

在。

= argmax 一个 Qt(a), (2.2)

其中 argmaxa 表示动作 a，后面的表达式将为其最大化 (同样，连接将被任意破坏)。贪婪的行为选择总是利用当前的知识以获得最大的即时回报; 它没有花任何时间抽样明显的劣等行为，看看它们是否真的更好。一个简单的替代方法是贪婪行为的大部分时间, 但每隔一段时间, 说有小概率的 , 相反

从所有具有相等概率的动作中随机选择，独立于动作值估计。我们调用方法使用此 near-greedy 行动选择定则 -greedy 方法。这些方法的一个优点是, 在极限的步数增加, 每一个动作将取样无限次数, 从而确保所有的 Qt(a) 收敛于问 ∗(a)。这当然意味着选择的概率收敛于最优行动大于 1− , 附近就是必然的。然而，这些只是渐进的保证，对这些方法的实际有效性只

字未提。

练习 2.1 -greedy 行动选择, 两个动作和 = 0.5 的情况下, 是什么选择贪婪行为的概率? ?

2.3 的 10-armed 试验台

粗略评估的相对有效性贪婪、 -greedy 行为价值方法, 我们比较数值在一套测试问题。这是一组随机生成的 k -武装土匪问题，k = 10。对于每一个土匪问题, 如一个如图 2.1 所示, 动作值, 问 ∗(a), 一个 = 1,。10

行动

图 2.1: 来自 10 臂测试床的一个 bandit 问题示例。真正价值 q∗(a) 这十个动作的每一个动作都是根据平均值为零和单位的正态分布来选择的方差, 然后选择实际奖励根据意思问 ∗(a) 单位方差正态分布, 这些灰色分布所显示。

2.3。10-armed 测试平台 29 日

根据均值为 0，方差为 1 的正态 (高斯) 分布进行选择。然后, 当一个学习方法应用到这个问题在 t 时间步选择行动, 实际的奖励,Rt, 选择从一个正态分布的意思是问 ∗(At) 和方差 1。这些分布如图 2.1 所示。我们把这组测试任务称为 10 臂测试床。对于任何一种学习方法，我们都可以度量它的性能和行为，当它应用于某个土匪问题时，它会随着时间的推移而改进。这是一次跑步。在 2000 次独立运行中，我们重复了这个过程，每个运行都有不同的班迪特问题，我们得到了学习算法的平均行为的度量。图 2.2 比较贪婪的方法与两个 -greedy 方法 (= 0.01 和 = 0.1), 如上所述, 在 10-armed 试验台。所有的方法都使用样本平均技术来形成它们的动作值估计。上面的图表显示了期望报酬随着经验的增加而增加。贪心方法在一开始比其他方法改进得稍快一些，但随后在较低的级别趋于平稳。它的每步奖励只有 1，而在这个测试台上，每步奖励的最高奖励是 1.55。贪婪方法在长期运行中表现得非常糟糕，因为它

步骤

步骤

图 2.2: 平均性能 -greedy 行为价值方法 10-armed 试验台。这些数据平均运行了 2000 次，并且有不同的盗版者问题。所有方法示例平均作为他们的行动价值估计。

经常被卡在执行次优的动作中。下面的图显示贪婪方法只在大约三分之一的任务中找到了最优动作。在另外三分之二的情况下，其最优行为的初始样本令人失望，而且它从未返回。 -greedy 方法最终表现更好, 因为他们继续探索和提高的机会识别最优行动。 = 0.1 方法探索更多, 通常发现最优行动之前, 但它从来没有选择的行动超过 91 -greedy 在贪婪的优势取决于任务的方法。例如，假设奖励方差更大，比如 10 而不是 1。与奖励吵着需要更多的探索发现最佳的行动, 和 -greedy 方法应该表现更好的相对于贪婪的方法。另一方面，如果奖励方差为零，那么贪心方法尝试一次后就会知道每个行为的真实值。在这种情况下，贪婪方法实际上可能表现得最好，因为它很快就会找到最优动作，然后永远不会探索。但即使是在确定性的情况下，如果我们弱化了其他假设，也会有很大的优势。例如，假设 bandit 任务是非平稳的，也就是说，动作的真实值随着时间的变化而变化。在这种情况下，即使在确定性的情况下，也需要进行探索，以确保其中一个非贪婪行为没有改变，使其变得比贪婪行为更好。正如我们将在接下来的几章中看到的，非平稳性是在强化学习中最常见的情况。即使底层任务是固定的和确定性的，学习者也面临一组类似班迪特的决策任务，每一项任务都随着学习的进展和代理的决策策略的变化而变化。强化学习需要探索和开发之间的平衡。练习 2.2: 土匪例子考虑一个 k 武装土匪问题，k = 4 个动作，表示 1、2、3 和 4。考虑申请这个问题一个赌博机算法使用 -greedy 行动选择, 样本企业平均行为价值的估计, 并初步估计 Q1(a)= 0, 所有。假设初始序列的行为和奖励是 A1 = 1,R1 = 1,A2 = 2,R2 = 1,A3 = 2,R3 = 2,A4 = 2,R4 = 2,A5 = 3,R5 = 0。在这些时间步骤 的情况可能发生, 导致一个动作是随机选择的。这在什么时候发生过? 在什么时间步可以做到这一点可能发生呢? ? 练习 2.3 在图 2.2 所示的比较中，从长期来看，哪种方法在累积奖励和选择最佳行为的概率方面表现最好? 会有多好? 定量地表达你

的答案。?

2.4 增量实现

到目前为止我们已经讨论过的行动-价值方法都是作为观察到的奖励的样本平均值来估计行动值。现在我们要讨论的问题是，如何以一种计算效率很高的方式来计算这些平均数，特别是用恒定的内存

和持续 per-time-step 计算。为了简化表示法，我们只关注一个动作。现在让 Ri 表示收到的奖励后第 i 个选择的行动, 并让 Qn 表示其动作值的估计后选择 n−1 次, 我们现在可以简单地写成 Qn。= R1 + R2 + ••• + Rn−1 n−1 。

显然的实现方法是保存所有奖励的记录，然后在需要估计值时执行此计算。但是，如果这样做了，那么随着时间的推移，内存和计算需求会随着时间的推移而增长。每个额外的奖励将需要额外的内存来存储它，并需要额外的计算来计算分子的和。正如您可能会怀疑的，这并不是真正必要的。我们很容易设计出增量公式来更新平均值，而处理每一个新的奖励都需要不断的小计算。给定 Qn 和第 n 个奖励 Rn，可以计算所有 n 个奖励的新平均值

Qn + 1 = 1 n n ?i = 1 国际扶轮

= 0 O

Rn + n−1 ?i = 1 国际扶轮

这在 n = 1 时成立，对于任意 Q1，得到 Q2 = R1。这个实现只需要 Qn 和 n 的内存，并且每个新的奖励只需要很小的计算量 (2.3)。这个更新规则 (2.3) 是在本书中经常出现的形式。一般的形式是

NewEstimate←OldEstimate + StepSize

目标 −OldEstimate

。(2.4)

表达式

目标 −OldEstimate

是估计误差。它通过服用而减少向“目标”迈出的一步。“虽然目标可能很吵，但它被认为是一个理想的移动方向。”例如，在上面的例子中，目标是第 n 个奖励。注意，增量方法 (2.3) 中使用的步长参数 (逐步大小) 会随着时间步长而变化。在处理第 n 项奖励措施 a 时

方法使用步长参数 1 n 。在本书中，我们表示步长参数由 或更普遍, t(a)。完整的赌博机算法的伪代码使用增量计算样本平均值和 -greedy 行为选择下面的框所示。函数 bandit(a) 被假设采取一个动作并返回相应的奖励。

2.5 跟踪非平稳问题

到目前为止所讨论的平均方法适用于固定的赌博机问题，也就是说，适用于赌博机问题，在这种问题中，报酬概率不会随时间变化。如前所述，我们经常遇到有效的非平稳的强化学习问题。在这种情况下，给予近期奖励而不是长期奖励更有意义。最常用的一种方法是使用常量步长参数。例如, 增量更新规则 (2.3) 的平均更新 Qn n−1 修改过去的奖励

Qn + 1。= Qn +

Rn−Qn

, (2.5)

在步长参数 (0,1) 是恒定的。这导致 Qn+1 是过去奖励和初始估计 Q1 的加权平均值:

Qn + 1 = Qn +

Rn−Qn

Qn = Rn +(1−) = Rn +(1−)[Rn−1 +(1−)Qn−1]= Rn +(1−) Rn−1 +(1−)2 Qn−1 = Rn +(1−) Rn−1 +(1−)2 Rn−2 + n ••• +(1−)−1 r1 +(1−)nQ1 =(1−)nQ1 + n ?i = 1 (1−)n−iRi。(2.6)

我们称之为加权平均, 因为重量的总和 (1−)n + ?n i = 1 (1−)n−i = 1, 你可以检查你自己。注意体重, (1−)n− 我给奖励 Ri 取决于有多少奖励前,n− 我观察到。数量 1− 小于 1, 从而给国际扶轮的重量减少干预奖励数量的增加。事实上, 体重指数衰减 −1 日根据指数 。(如果 1− = 0, 那么所有的重量最后奖励,Rn, 因为公约 00 = 1)。因此，这有时被称为指数加权平均。有时步长参数的变化很方便。让 n(a) 表示步长参数用于处理接收到的奖励后 n 选择行动的。正如我们所指出的那样, 选择 n(a)= 1 n 结果样本均值的方法, 这是保证收敛于真实的行动值由大数定律。当然收敛并不能保证对所有选择的序列 n(a)。随机逼近理论中一个众所周知的结果为我们提供了保证具有概率 1 的收敛性所需的条件:

∞ 吗?n = 1 n(a)=∞ ∞ 吗?n = 1 2 n(a)<∞。(2.7)

第一个条件是保证步骤足够大，最终能够克服任何初始条件或随机波动。第二个条件保证最终步骤足够小，以确保收敛。注意, 两个样品平均情况下满足收敛条件, n(a)= 1 n, 但不是为步长参数不变的情况下, n(a)= 。在后一种情况下，第二个条件没有满足，这表明估计数从不完全收敛，而是继续随最近收到的奖励而变化。正如我们上面提到的，这在非平稳环境中是可取的，有效的非平稳问题在强化学习中是最常见的。此外，满足 (2.7) 条件的步长参数序列通常收敛得很慢，或者需要进行大量的调优才能获得满意的收敛速度。虽然满足这些收敛条件的步长参数序列常用于理论研究，但在实际应用和实证研究中很少用到。练习 2.4 如果步长参数、 n 不是常数, 那么估计 Qn 之前收到的奖励是一个加权平均权重不同, 由 (2.6) 给出。类似地，一般情况下，每个事先奖励的权重是多少对于 (2.6)，根据步长参数序列? ? 练习 2.5(编程) 设计并进行了一个实验来演示样本平均方法在非平稳问题上的困难。使用一种修改版的 10-armed 试验台所有 q∗(a) 开始平等, 然后采取独立的随机漫步 (比如通过添加一个正态分布与平均零和标准偏差 0.01 增加所有问 ∗(a) 在每一个步骤)。准备图如图 2.2 所示的行为价值使用样本平均方法, 逐步计算, 和另一个行为价值的方法使用一个常数步长参数 = 0.1。使用 = 0.1 和长跑,10000 步说。?

2.6 乐观的初始值

到目前为止，我们讨论的所有方法在某种程度上都依赖于初始的行动值估计 Q1(a)。在统计学的语言中，这些方法由于最初的估计而有偏差。样品平均方法, 偏见消失一旦所有行动至少有一次被选中, 但与常数 为方法, 偏差是永久性的, 但随着时间减少由 (2.6) 给出。在实践中，这种偏见通常不是问题，有时会很有帮助。缺点是初始估计实际上变成了用户必须选择的一组参数，如果只是将它们全部设置为 0 的话。好处是，它们提供了一种简单的方法，可以提供一些关于预期奖励水平的预先知识。初始动作值也可以用作鼓励探索的简单方法。假设我们没有将初始动作值设置为 0，而是将它们全部设置为 +5。记得问 ∗(a) 在这个问题上选择从一个正态分布均值为 0, 方差为 1。因此，对 +5 的初步估计非常乐观。但这种乐观鼓励探索具有行动价值的方法。无论最初选择什么行动，奖励都小于初始估计; 学习者转向其他行为，对所得到的奖励感到“失望”。结果是，所有操作都在值估计收敛之前进行了多次尝试。即使总是选择贪婪的行为，系统也会进行大量的探索。图 2.3 显示了性能 10-armed 赌博机试验台的贪婪的方法使用 Q1(a)= + 5, 一。相比之下, 也显示是一个 -greedy 方法 Q1(a)= 0。起初，乐观的方法表现得更差，因为它探索得更多，但最终它表现得更好，因为它的探索随着时间的推移而减少。我们称这种鼓励勘探的技术为乐观初始值。我们认为它是一种简单的技巧，可以在固定的问题上非常有效，但它远不是鼓励探索的一种普遍有用的方法。例如，它不太适合非平稳问题，因为它的探索动力是内在的

戏剧的一步

图 2.3: 乐观的初始行动价值估计对 10 个武装试验台的影响。这两种方法都使用一个常数

步长参数, = 0.1。

2.7 Upper-Confidence-Bound 选择动作

暂时的。如果任务发生变化，产生了对探索的新需求，那么这种方法就没有帮助了。实际上，任何以任何特殊方式集中于初始条件的方法都不太可能帮助处理一般的非平稳情况。时间的开始只发生一次，因此我们不应该过多地关注它。这种批评同样适用于抽样平均方法，它也将时间的开始视为一个特殊的事件，平均所有后续的奖励都是相同的权重。然而，所有这些方法都非常简单，其中的一种——或者是它们的简单组合——通常在实践中是足够的。在本书的其余部分，我们经常使用这些简单的探索技巧。练习 2.6: 神秘的尖峰图 2.3 中显示的结果应该是相当可靠的，因为它们是超过 2000 个人的平均值，随机选择 10 个武装的土匪任务。那么，为什么乐观方法在曲线的早期会出现振荡和峰值呢? 换句话说，是什么让这个方法表现得更好更糟的是，在早期阶段? ? 练习 2.7: 无偏恒步长技巧在本章的大部分内容中，我们使用样本平均来估计动作值，因为样本平均不会产生常数步长所产生的初始偏差 (参见 (2.6) 的分析)。然而，样本平均值并不是一个完全令人满意的解决方案，因为它们可能在非平稳问题上表现不佳。是否有可能避免步长恒定的偏差，同时保留它们在非平稳问题上的优势? 一种方法是使用步长。

n。= / ōn, (2.8) 处理 n 奖励一个特定的行动, 其中 > 0 是一个传统的固定步长, 和 ōn 是一个从 0 开始的跟踪:

ōn 。= ōn−1 + (1−ōn−1), 与 ōn 0,0。= 0。(2.9)

在 (2.6) 中进行这样的分析，以表明 Qn 是指数加权的平均没有最初的偏见。?

2.8 pper-Confidence-Bound 选择动作

需要进行探索，因为对行动价值估计的准确性总是存在不确定性。贪婪的行为是那些目前看起来最好的行为，但是其他的一些行为实际上可能更好。 -greedy 选择动作迫使贪婪的行为进行审判, 但不加区别地, 几乎没有偏爱那些贪婪的或特别不确定。在非贪婪行为中，根据它们实际上是最优行为的潜力进行选择是更好的，要考虑到它们的估计离最大值有多近以及这些估计中的不确定性。这样做的一个有效方法是根据以下内容选择操作

ln t 表示的自然对数 (e 2.71828 数量必须提高到为了 = t),Nt(a) 表示, 行动已经被选择的次数在时间 t(2.1)(分母), 和 c > 0 数量控制程度的探索。如果 Nt(a) = 0，则 a 被认为是最大化行为。上置信区间 (UCB) 作用选择的概念是平方根项是对 a 值估计的不确定性或方差的度量。因此，被最大值覆盖的量是作用 a 的可能真值的上界，c 决定置信水平。每次选 a 时，不确定性大概会减少:Nt(a) 增量，并且，当它出现在分母中，不确定项就会减少。另一方面，每一次除了 a 以外的动作，t 增加，但 Nt(a) 不增加; 因为 t 出现在分子中，所以不确定性估计值增加。自然对数的使用意味着随着时间的推移增加会越来越小，但是是无界的; 所有操作最终都将被选中，但是值估计值较低的操作，或者已经频繁地被选中的操作，将会随着时间的推移而逐渐减少。结果与 UCB 在 10 个武装试验床上显示在图 2.4。UCB 通常表现良好, 如下所示, 但是更困难比 -greedy 超越赌博机更一般的强化学习设置在这本书的其余部分。一个困难是处理非平稳问题; 将需要比第 2.5 节中所介绍的方法更为复杂的方法。另一个困难是处理大的状态空间，特别是在使用函数逼近时，如本书第二部分所述。在这些更高级的设置中，UCB 动作选择的想法通常是不实际的。

图 2.4:10 臂试验台 UCB 动作选择的平均性能。如图所示, UCB 一般执行比 -greedy 选择动作, 除了在第一个 k 步骤, 当它选择随机 as-yet-untried 行动之一。

练习 2.8:UCB 峰值在图 2.4 中 UCB 算法显示了在第 11 步上性能的明显峰值。这是为什么呢? 注意，为了让你的回答完全令人满意，它必须解释为什么奖励在第 11 步增加，为什么它在随后的步骤减少。提示: 如果 c = 1，那么峰值就不那么明显了。?

2.9 梯度赌博机算法

到目前为止，在这一章中，我们已经考虑了估计动作值的方法，并使用这些估计来选择动作。这通常是一个很好的方法，但并不是唯一可行的方法。在本节中，我们考虑学习对每个动作 a 的数值偏好，我们将其命名为 Ht(a)。偏好越大，采取行动的频率就越高，但这种偏好在回报方面没有解释。只有一种行动相对于另一种行动的相对偏好是重要的; 如果我们将 1000 加到所有的操作偏好中，则不会对操作概率产生影响，操作概率是根据一个软最大值分布 (即，吉布斯或波尔兹曼分布) 如下:

在这里我们还引入了一个有用的新符号, t(a), 采取行动的概率在时间 t。最初所有的行动偏好是相同的 (例如,H1(a)= 0, 所有 a), 所有的行动都有一个相同的概率被选中。练习 2.9 表明，在两种行为的情况下，软最大值分布与经常用于统计和人工的 logistic(或 sigmoid) 函数的分布相同神经网络。?

基于随机梯度上升的思想，本文提出了一种自然学习算法。在每一步中，在选择动作并收到奖励 Rt 后，动作偏好更新为:

Ht + 1(在)。= Ht(在)+ ?Rt−Rt ? ?1− t(在)?, Ht + 1(a)。= Ht(一)− ?Rt−Rt ? t(a), 对所有? =,(2.12) > 0 是一个步长参数,t R 是平均水平的回报, 包括时间 t, 可以计算增量如 2.4 节所述 (如果问题是不稳定或第 2.5 节)。Rt 项作为基线的奖励比较。如果奖励高于基线，那么未来接受 At 的概率就会增加，如果奖励低于基线，那么概率就会降低。非选择的动作则朝相反的方向移动。图 2.5 显示了梯度班迪特算法对 10 臂测试床的一个变体的结果，在这个测试床中，真正的期望奖励是根据平均值为 +4 而不是 0 的正态分布 (和以前一样，单位方差) 来选择的。由于奖励基线条件的存在，所有奖励的上升对梯度班迪特算法没有任何影响，它会立即适应新的水平。但如果基线是省略了 (也就是说, 如果 Rt 是常数零 (2.12)), 则性能会明显退化, 如图。

80 年

最佳的行动 60

40

20.

步骤

图 2.5: 平均梯度赌博机算法的性能与和没有回报基线 10-armed 试验台时间 ∗+ 4 附近 (a) 被选择而不是接近于零。

2.10 关联搜索 (上下文盗匪)

到目前为止，在这一章中，我们只考虑了非关联任务，也就是说，不需要将不同的操作与不同的情况关联起来的任务。在这些任务中，学习者要么试图在任务静止时找到一个最佳的动作，要么在任务不稳定时试图跟踪最佳的动作。然而，在一般的强化学习任务中有不止一种情况，目标是学习一种策略: 从一种情况映射到在那种情况下最好的行动。为了为整个问题设置好舞台，我们简要地讨论了非关联任务扩展到关联设置的最简单方式。举个例子，假设有几个不同的 k 武装的土匪任务，在每个步骤中你都随机地遇到其中的一个。因此，bandit 任务从一个步骤随机地变化。在您看来，这是一个单一的、非平稳的 k 武装土匪任务，其真正的动作值从一步到一步随机变化。您可以尝试使用本章中描述的方法之一来处理非平稳性，但是除非真正的动作值变化缓慢，否则这些方法不会很好地工作。但是，现在假设，当为您选择一个 bandit 任务时，您会得到一些关于它的标识的独特线索 (但不是它的操作值)。也许你面对的是一台真正的老虎机，当它改变其动作值时，它会改变显示的颜色。现在，您可以学习一种策略，将每个任务 (用您看到的颜色表示) 与面对任务时要采取的最佳行动相关联——例如，如果是红色，选择 arm 1; 如果是绿色，选择 arm2。有了正确的策略，您通常可以做得比没有任何信息区分一个班迪特任务和另一个班迪特任务要好得多。这是一个关联搜索任务的例

子，之所以叫它，是因为它包含了尝试和错误学习来寻找最佳行为，以及这些行为与最佳情况的关联。联想搜索任务现在经常被称为文献中的上下文赌博机。关联搜索任务介于 k -武装匪徒问题和充分强化学习问题之间。它们就像完整的强化学习问题因为它们涉及到学习一个策略，但是就像我们版本的 k 武装匪徒问题一样，每个动作只影响即时的奖励。如果行为被允许影响下一个情境以及奖励，那么我们就有了完全强化学习的问题。我们将在下一章中介绍这个问题，并考虑它在本书其余部分的影响。

练习 2.10 假设你面对一个 2 臂的土匪任务，它的真实动作值会随着时间的推移而随机变化。具体来说, 假设, 对于任何时间步, 行动的真实值 1 和 2 分别为 0.1 和 0.2 的概率 (情况下),0.5,0.9 和 0.8 的概率 0.5(B)。如果你不能告诉你的脸在任何一步, 成功的最好的期望是什么, 你可以实现, 你应该如何实现? 现在假设在每一步你都被告知你面对的是情形 A 还是情形 B(尽管你仍然不知道真正的行为值)。这是一个关联搜索任务。在这方面，你能达到的最大成功期望是什么任务，你应该怎么做才能完成？？

2.11 总结

在本章中，我们介绍了几种平衡勘探和开发的简单方法。 -greedy 方法选择随机的一小部分, 而 UCB 选择确定性但实现勘探的方法巧妙地支持在每一步的行动到目前为止收到较少的样本。梯度班迪特算法不是估计动作值，而是估计动作偏好，并倾向于使用软最大值分布以一种分级的、概率的方式来估计更喜欢的动作。乐观地初始化估计的简单权宜之计导致甚至贪婪的方法也要进行大量的探索。人们自然会问，哪种方法是最好的。虽然这是一个很难回答的问题，但我们可以在我们在本章中使用的十臂测试台上进行测试，并比较他们的表现。复杂的是它们都有一个参数; 为了得到一个有意义的比较，我们必须考虑它们的性能作为参数的函数。到目前为止，我们的图已经显示了每个算法和参数设置的学习过程，为该算法和参数设置生成一个学习曲线。如果我们为所有的算法和所有的参数设置绘制学习曲线，那么图就会太复杂和拥挤，无法进行清晰的比较。相反，我们总结一个完整的学习曲线，用它的平均值除以 1000 步; 这个值与学习曲线下的面积成正比。图 2.6 显示了本章中各种班迪特算法的度量，每个都作为自己的参数的函数，在 x 轴上的单个标度上显示。这种图称为参数研究。注意，参数值是根据两个因子的不同而变化的，并以对数尺度表示。还注意每种算法性能的特征逆变 u 型; 所有算法在其参数的中间值上都表现得最好，既不太大也不太小。在评估

c Q0 图 2.6: 本章给出的各种土匪算法的参数研究。每一个点都是在特定的参数设定下，获得超过 1000 步的平均奖励。

作为一个方法，我们不仅要关注它在最佳参数设置中的表现，还要关注它对参数值的敏感性。所有这些算法都是相当不敏感的，在一个数量级的参数值范围内表现良好。总的来说，在这个问题上，UCB 似乎表现最好。尽管这些方法很简单，但我们认为这一章介绍的方法可以被认为是最先进的。有更复杂的方法，但是它们的复杂性和假设使它们不适合我们真正关注的完全强化学习问题。从第五章开始，我们提出了解决全强化学习问题的学习方法。虽然本章探讨的简单方法可能是目前我们所能做的最好的方法，但对于平衡勘探和开发的问题，这些方法还远远不能完全令人满意的解决办法。在 k -armed bandit 问题中，平衡勘探和开发的一个很好的方法是计算一种称为 Gittins 指数的特殊行为值。在某些重要的特殊情况下，这种计算是可处理的，并直接导致最优解，尽管它确实需要对可能的问题的先验分布有完整的了解，我们通常认为这是不可用的。此外，这种方法的理论和计算可追溯性似乎都不能推广到我们在本书其余部分中考虑的完全强化学习问题。gittin -index 方法是贝叶斯方法的一个实例，它假设在操作值上有一个已知的初始分布，然后在每一步之后更新这个分布 (假设真正的操作值是平稳的)。通常，更新计算非常复杂，但是对于某些特殊的分布 (称为共轭先验)，它们很容易。一种可能性是，根据每个步骤的后验概率选择最佳动作。这种方法，有时称为后验抽样或汤普森抽样，通常与我们在本章中所介绍的最佳无分布方法相似。在贝叶斯条件下，甚至可以计算出勘探和开发之间的最优平衡。一个人可以计算任何可能的行动的概率每个可

能的即时奖励和结果的后验分布超过行动值。这种不断发展的分布成为问题的信息状态。给定一个范围，比如说 1000 步，你可以考虑所有可能的行动，所有可能的结果奖励，所有可能的下一步行动，所有的下一个奖励，等等所有的 1000 步。在假设的前提下，每一个可能的事件链的回报和概率都是可以确定的，人们只需要选择最好的。但是可能性之树生长得非常快; 即使只有两个动作和两个奖励，这棵树也会有 22000 片叶子。准确地进行这种巨大的计算通常是不可行的，但也许可以有效地近似。这种方法将有效地将土匪问题转化为完全强化学习问题的一个实例。最后，我们可以使用近似强化学习方法，如本书第二部分所介绍的方法，来接近这个最优解。但这是一个研究的话题，超出了这本入门书的范围。

练习 2.11(编程) 使图与练习 2.5 中列出的非平稳情况的图 2.6 相似。包括 constant-step-size -greedy 算法与 $= 0.1$。使用 20 万步的运行，作为每个算法和参数设置的性能度量，使用最后 10 万步的平均奖励。?

书目的和历史的言论

2.1 在统计学、工程、心理学等方面对土匪问题进行了研究。在统计，赌博机问题属于“经验的顺序设计”，由汤普森 (1933,1934) 和罗宾斯 (1952) 介绍，贝尔曼 (1956) 研究。Berry 和 Fristedt(1985) 从统计学的角度对赌博机问题进行了广泛的处理。纳伦德拉 (Narendra and Thathachar)(1989) 从工程的角度来处理土匪问题，很好地讨论了针对土匪的各种理论传统。在心理学上，班迪特问题在统计学习理论中发挥了作用 (如 Bush 和 Mosteller, 1955; 埃斯蒂斯,1950)。在启发式搜索文献中经常使用“贪心”一词 (例如 Pearl, 1984)。在控制工程中，勘探和开发之间的冲突被称为识别 (或估计) 和控制之间的冲突 (例如，Witten, 1976b)。Feldbaum(1965) 将其称为双控问题，指在不确定性控制系统时需要同时解决识别和控制两个问题。在讨论遗传算法的各个方面时，Holland(1975) 强调了这种冲突的重要性，并将其称为需要利用和需要新信息之间的冲突。

2.2 针对我国 k 武装土匪问题提出的行动价值方法 Thathachar 和 Sastry(1985)。在学习自动机文献中，这些通常被称为估计算法。动作值一词源于 Watkins(1989)。首先使用 -greedy 方法也可能被沃特金斯 (1989, 第 187 页), 但这个想法非常简单, 一些早期使用可能。

2.4-5 该材料属于随机迭代算法的一般标题，这些都被 Bertsekas 和 Tsitsiklis(1996) 所涵盖。

2.6 Sutton(1996) 在增强学习中使用了乐观初始化。

2.7 早期的工作是使用估计的上置信值来选择动作由 Lai and Robbins(1985)、Kaelbling (1993b) 和 Agrawal(1995) 完成。我们在这里展示的 UCB 算法在文献中被称为 UCB1，最初由 Auer、Cesa-Bianchi 和 Fischer(2002) 开发。

2.8 梯度土匪算法是基于梯度的强化的一个特例由 Williams(1992 年) 引入的学习算法，后来发展成为我们在本书后面所讨论的针对演员和政策梯度的算法。我们在这里的发展受到巴拉曼 • 拉文德拉 (个人) 的影响

沟通)。格林史密斯 (Greensmith)、巴特莱特 (Bartlett)、巴克斯特 (Baxter, 2002, 2004) 和迪克 (Dick)(2015) 提供了关于基线选择的进一步讨论。萨顿 (1984) 对这类算法进行了早期的系统研究。动作选择规则 (2.11) 的“软最大值”一词源于《笼头》(1990)。这条规则似乎是卢斯 (1959) 首先提出的。

2.9 提出了联想搜索这一术语及其对应的问题 Barto, Sutton 和 Brouwer(1981)。联合强化学习这个术语也被用于联合搜索 (Barto 和 Anandan, 1985)，但是我们更倾向于将这个术语作为充分强化学习问题的同义词 (如 Sutton, 1984)。(正如我们所指出的，现代文学也用“语境土匪”这个词来描述这个问题。) 我们注意到桑代克的效应定律 (在第一章中引用) 描述了关联搜索，它指的是情景 (状态) 和行为之间的关联关系的形成。根据操作性条件作用或工具性条件作用的术语 (如斯金纳，1938)，区别性刺激是一种指示特定强化偶然性存在的刺激。用我们的话说，不同的辨别刺激对应不同的状态。

2.10 Bellman(1956) 是第一个展示如何使用动态编程的在贝叶斯公式中计算勘探开发的最优平衡问题。Gittins 索引方法是由 Gittins 和 Jones(1974) 提出的。Duff(1995) 展示了如何通过强化学习来学习赌博机问题的 Gittins 指数。Kumar(1985) 的调查很好地讨论了贝叶

斯方法和非贝叶斯方法。“信息状态”一词来源于关于部分可观测的 MDPs 的文献; 见, 例如, 洛夫乔伊 (1991)。其他的理论研究集中在探索的效率上，通常表现为算法能以多快的速度接近最优的决策策略。确定搜索效率的一种方法是适应增强学习一种监督学习算法的样本复杂度的概念，这是算法在学习目标函数时需要的训练实例数量。对一个增强学习算法的探索样本复杂性的定义是算法没有选择接近最佳动作的时间步骤 (Kakade, 2003)。Li(2012) 在研究增强学习中探索效率的理论方法时，讨论了这一方法和其他几种方法。Russo、Van Roy、Kazerouni、Osband 和 Wen(2018) 对汤普森采样进行了全面的现代处理。

3. 第三章有限马尔可夫决策过程

在本章中，我们将介绍有限的马尔可夫决策过程的形式问题，或有限的 MDPs，我们试图在本书的其余部分中加以解决。这个问题涉及到评价反馈，如在土匪中，但也包括在不同情况下选择不同行动的联想方面。MDPs 是顺序决策的典型形式化，在这种情况下，行动不仅会影响即时奖励，还会影响随后的情况或状态，并通过这些未来的奖励。因此，千年发展目标包括延迟奖励和权衡即时和延迟奖励。我们估计的价值而在土匪问题问 ∗(a) 的每一个行动, 在 mdp 我们估计价值问 ∗(,) 的每一个行动在每个州年代, 或我们估计价值 v∗(s) 的每个状态最优行动选择。这些与国家有关的量对于准确地为个人行为选择的长期后果分配信用至关重要。MDPs 是一种数学上理想的强化学习问题形式，可以对其进行精确的理论表述。我们介绍了问题数学结构的关键要素，如返回、值函数和 Bellman 方程。我们试图传达广泛的应用，这些应用可以被表述为有限的 MDPs。在所有的人工智能中，适用性的广度和数学可追溯性之间存在着一种张力。在本章中，我们将介绍这种紧张关系，并讨论它所包含的一些权衡和挑战。在第 17 章中讨论了可以在多边开发计划署之外进行强化学习的一些方法。

3.1 Agent-Environment 接口

千年发展目标应该是对从交互中学习以实现目标的问题的一个简单的框架。学习者和决策者被称为代理人。它与之交互的东西，包括代理之外的所有东西，叫做环境。这些交互是连续的，代理选择动作，环境响应

48 第三章: 有限马尔可夫决策过程

这些行为和向代理提供新情况。环境也会产生回报，这种特殊的数值是代理人通过选择行动寻求最大化的。

图 3.1: 马尔可夫决策过程中的代理-环境交互。

更具体地说，代理和环境在每个离散时间步骤的序列中相互作用，t = 0, 1, 2, 3，⋯2 在每个时间步 t, 代理接收环境的的一些表示状态, 圣　年代, 并在此基础上选择一个动作, (S)。3 一个时间步后, 部分作为其行动的后果, 代理接收到一个数值奖励,Rt + 1　R, 并发现自己在一个新的国家, 圣 + 1.4 MDP 和代理在一起从而产生一个序列或轨迹, 开头是这样的:

S0 A0 R1 S1 A1 R2 S2 A2 R3⋯(3.1)

在一个有限的 MDP 中，状态、行为和奖励 (S、a 和 R) 的集合都有有限数量的元素。在这种情况下，随机变量 Rt 和 St 已经定义好了仅依赖于前一状态和行为的离散概率分布。也就是说，对于这些随机变量 s 的特定值? 年代和　r, 这些值的概率发生在时间 t, 鉴于特定值前状态和行动:

p(s ?，r |s a) 圣 = = 公关年代?,Rt = r | 圣 −1 = s−1 = , (3.2)

对于所有年代? 年代, r, (s)。函数 p 定义了 MDP 的动力学。等式中等号两边的点提醒我们，它是一个定义 (在这个例子中是函数 p)，而不是遵循先前定义的事实。动态函数 p:S×R×S×→[0,1] 是一个普通的确定性函数的四个参数。中间的 | 来自条件概率的符号，

我们使用术语代理、环境和操作，而不是工程师的术语控制器、控制系统 (或工厂) 和控制信号，因为它们对更广泛的用户来说是有意义的。我们将注意力限制在离散时间上，以使事情尽可能地简单，即使许多想法可以扩展到连续时间情况 (例如，参见 Bertsekas 和 tsiklis, 1996; 没有事情,1996)。为了简化表示法，我们有时假设在所有状态下动作集相同的特殊情况，并将其简单地写成 A。4 我们用 Rt+1 而不是 Rt 来表示由于 At 引起的奖励，因为它强调下一个奖励和下一个状态，Rt+1 和 St+1 是共同确定的。不幸的是, 这两个公约在文献中广泛使用。

但这里它只是提醒我们 p 指定了每个选项 s 和 a 的概率分布，也就是

s ? 年代

r p(s ?)= 1,r | 年代, 年代, (s)。(3.3)

在马尔可夫决策过程中，p 给出的概率完全表征了环境的动态。, 每个可能值的概率为 St 和 Rt 只取决于立即前状态和行动, 圣在 −−1 和 1, 早些时候给他们, 而不是在所有州和行动。这是最好的限制，不是在决策过程中，而是在状态上。该状态必须包含有关过去的代理-环境交互的所有方面的信息，这些交互将对未来产生影响。如果有，那么状态就有马尔可夫性质。我们将在整本书中假设马尔可夫性质，尽管从第二部分开始我们将考虑不依赖于它的近似方法，在第 17 章我们将考虑如何从非马尔可夫观察中学习和构造马尔可夫状态。从 four-argument 动力学函数,p, 一个可以计算任何一个想了解环境, 如状态转换概率 (我们表示轻微的虐待的符号, 作为一个 three-argument 函数 p:S×× 一 →[0,1]),

p(s ?| 年代,a)。圣 = = 公关年代? 圣 −1 = s | =−1 =

r p(s ?，r |s, a)。(3.4)

我们也可以计算出预期回报政府行动对双参数函数 r:S×→r:

r(年代)。= E[圣 −1 = s Rt | 在 −1 = =

r R

s ? 年代 p(s ?，r |s a)，(3.5)

和预期回报 state-action-next-state 三元组作为 three-argument 函数 r:S××S→r,

r(s,s)。= E[圣 −1 = s Rt |−1 = 一个圣 = s ?]=

r R p(s ?r | s,) p(s ?| 年代,a)。(3.6) 在这本书中，我们通常使用四个参数的 p 函数 (3.2)，但是这些其他的符号有时也很方便。MDP 框架是抽象和灵活的，可以用许多不同的方式应用于许多不同的问题。例如，时间步骤不需要参考固定的实时时间间隔; 它们可以指任意连续的决策和行动阶段。这些动作可以是低级的控制，比如应用于机器人手臂马达上的电压，也可以是高级的决定，比如是否去吃午饭或去读研究生。同样，各州可以采取各种形式。它们可以完全由低层次的感觉 (如直接的传感器读数) 决定，也可以是更高层次和更抽象的感觉 (如对房间内物体的象征性描述)。构成一种状态的一些因素可能是基于对过去感觉的记忆

甚至完全是心理或主观的。例如，代理可能处于不确定对象在哪里的状态，或者在某种明确定义的意义上感到惊讶的状态。类似地，有些动作可能完全是脑力或计算能力的。例如，一些操作可能控制代理选择考虑的内容，或者它关注的地方。总的来说，行动可以是我们想要学习的任何决定，而国家可以是我们知道的任何可能有用的东西。特别是，agent 与环境之间的边界通常不同于机器人或动物身体的物理边界。通常情况下，边界会更接近于代理。例如，机器人的马达和机械连接及其传感硬件通常应该被视为环境的一部分，而不是 agent 的一部分。同样，如果我们将 MDP 框架应用于人或动物，肌肉、骨骼和感觉器官应该被视为环境的一部分。奖励也可能是在自然和人工学习系统的身体内部计算出来的，但被认为是在主体外部。我们遵循的一般规则是，任何不能被代理任意改变的事物都被认为是在它之外的，因此它是环境的一部分。我们不会假设环境中的所有东西都是未知的。例如，代理通常知道很多关于它的奖励是如何作为它的行为和它们所采取的状态的函数来计算的。但是我们总是认为

奖励计算是在代理之外的，因为它定义了面向代理的任务，因此必须超出其任意更改的能力。事实上，在某些情况下，代理可能知道它的环境是如何工作的，并且仍然面临一个困难的强化学习任务，就像我们可能知道像 Rubik 的立方体一样的难题是如何工作的，但是仍然不能解决它。代理环境边界代表代理绝对控制的极限，而不是其知识的极限。agent-environment 边界可以位于不同的位置，用于不同的目的。在一个复杂的机器人中，许多不同的代理可能同时运行，每个代理都有自己的边界。例如，一个代理可以做出高层决策，这些决策是由执行高层决策的低级代理所面对的状态的一部分。在实践中，一旦选择了特定的状态、行为和奖励，并确定了感兴趣的特定决策任务，就确定了代理环境的边界。MDP 框架是对目标导向学习问题的一个相当抽象的概念。提出任何感官的细节, 内存, 和控制装置, 以及任何一个目的是试图实现, 任何问题的学习目标导向行为可以减少到三个信号之间来回传递一个代理及其环境: 一个信号代表的选择由代理 (行动), 一个信号来表示的基础上选择是由 (美国), 和一个信号来定义代理的目标 (奖励)。这个框架可能不足以代表所有的决策学习问题，但是它已经被证明是非常有用和适用的。当然，每个任务的特定状态和操作都有很大的不同，它们的表示方式会对性能产生很大的影响。在强化学习中，就像在其他类型的学习中一样，这种表征性的选择比科学更具有艺术性。

在这本书中，我们提供了一些关于表示状态和行为的好方法的建议和例子，但是我们的主要重点是在选择了表示之后学习如何行为的一般原则。

例子 3.1: 生物反应器假设正在应用强化学习来确定生物反应器 (用于生产有用化学物质的大量营养和细菌) 的每时每刻的温度和搅拌速度。这种应用程序的操作可能是将目标温度和目标搅拌速率传递给较低水平的控制系统，从而直接激活加热元件和电机以达到目标。这些状态可能是热电偶和其他感官读数，可能是经过过滤和延迟的，加上表示容器中的成分和目标化学品的符号输入。回报可能是对生物反应器产生有用化学物质的速率的逐时刻测量。注意，这里每个状态都是传感器读数和符号输入的列表或向量，每个动作都是由目标温度和搅拌速度组成的向量。强化学习任务的典型特征是具有这种结构化表示形式的状态和动作。另一方面，奖励总是单一的数字。

例 3.2: 在重复的拾取和放置任务中，拾取和放置机器人考虑使用增强学习控制机器人手臂的运动。如果我们想要学习快速流畅的运动，学习代理将必须直接控制电机，并对机械连杆的当前位置和速度有低延迟的信息。这种情况下的动作可能是在每个关节上应用于每个电动机的电压，而状态可能是关节角度和速度的最新读数。每一个被成功拾取并放置的物体的奖励可能是 +1。为了鼓励平稳的运动，在每一步上，一个小的，消极的奖励可以作为运动的每一刻的“摇摆”的函数。

练习 3.1 设计适合 MDP 框架的三个示例任务，确定每个任务的状态、行为和奖励。让这三个例子尽可能地不同。该框架是抽象和灵活的，可以用许多不同的方式应用。至少在你的一个例子中以某种方式扩展它的极限。?

练习 3.2 是充分有效地表示所有目标导向的 MDP 框架学习任务? 你能想到任何明显的例外吗? ?

练习 3 考虑开车的问题。你可以用加速器、方向盘和刹车来定义动作，也就是你的身体与机器的交汇处。或者你可以更深入地定义它们——比如，橡胶和路面的交汇处，考虑到你的动作是轮胎扭矩。或者你可以进一步定义它们，比如，你的大脑和你的身体在哪里，肌肉抽搐来控制你的四肢。或者你可以去到一个非常高的水平，说你的行动是你开车的选择。什么是正确的级别，什么是在代理和环境之间划清界限的正确位置? 在什么基础上，线路的一个位置优于另一个? 选择一个地点而不是另一个地点有什么根本原因吗? 或者这是一个自由的选择??

3.2 目标和奖励

练习 3.4 给出了一个类似于例子 3.3 的表，但是对于 p(s?r |s a) 它应该有 s a s s s 的列?、r 和 p(s ?，r |s a)，每 4 个元组对应一行 p(s ?，r |s a) > 0。?

3.2 目标和奖励

在强化学习中，主体的目的或目标是通过一种特殊的信号被形式化的，称为奖励，从环境传递给主体。在每个时间步, 奖励是一个简单的数字,Rt r . 非正式代理的目标是最大化回报它收到的总量。这就意味着不能立即获得最大的回报，而是长期累积的回报。我们可以把这个非正式的想法明确地表述为奖励假设:

我们所说的目标和目的都可以被认为是一个接收到的标量信号 (称为奖励) 的累积和的期望值的最大值。

利用奖励信号将目标的概念形式化是强化学习最显著的特征之一。虽然从奖励信号的角度来制定目标一开始可能显得有限，但在实践中证明它是灵活和广泛适用的。看到这一点的最好方法是考虑如何使用它的例子。例如，为了让机器人学会走路，研究人员提供了与机器人向前运动成正比的每一步奖励。在制作一个机器人学习如何逃离迷宫, 奖励通常是 −1 每一个时间步, 经过之前逃离; 这鼓励代理尽可能快地转义。为了让机器人学会寻找和收集空的汽水罐回收利用，人们可能会在大部分时间给它一个零奖励，然后每个人都可以得到一个 +1 的奖励。当机器人撞到东西或者有人对它大喊大叫的时候，你可能也想给它一些负面的奖励。代理学习下棋或国际象棋, 自然赢得奖励 + 1, 失去 −1,0 为图纸和所有非终结符的位置。你可以看到在这些例子中发生了什么。代理人总是学会使其报酬最大化。如果我们想让它为我们做点什么，我们必须给它提供奖励，这样在最大化它们的同时，代理也会实现我们的目标。因此，至关重要的是，我们设立的奖励真正表明了我们想要实现的目标。

特别地，奖励信号不是向代理传授如何实现我们想要它做的事情的先验知识的地方。例如，一个下棋的代理人应该只因为他确实赢了而得到奖励，而不是因为他完成了子目标，如夺取对手的棋子或控制棋盘的中心。如果实现这些子目标得到了奖励，那么代理可能会找到一种方法来实现它们，而不会实现真正的目标。例如，它可能会找到一种方法来夺取对手的棋子，即使是以输掉比赛为代价的。奖励信号是你向机器人传达你想让它实现什么的方式，而不是你想让它实现的方式

3.3 回报和集

到目前为止，我们已经讨论了非正式学习的目标。我们已经说过，代理人的目标是使其长期获得的累积回报最大化。这是如何正式定义的? 如果时间步 t 后的奖励序列为 Rt+1, Rt+2, Rt+3，⋯，那么这个序列的具体方面是什么呢? 一般来说，我们寻求的是期望收益最大化，其中的收益 (表示 Gt) 被定义为奖励序列的某个特定函数。在最简单的情况下，回报是回报的总和:

Gt。= Rt+1 +Rt+2 +Rt+3 + ••• + Rt，(3.7)

T 是最后一步。这种方法在应用程序中是有意义的，在这些应用程序中，有一个自然的时间步概念，即当代理环境交互自然地分解为子序列时，我们将其称为“插曲”，如游戏中的游戏，在迷宫中旅行，或任何类型的重复交互。每一集都以一种特殊的状态结束，这种状态被称为终端状态，然后重置为标准启动状态，或者从标准启动状态分布返回到样本。即使你认为每一集都以不同的方式结尾，比如输赢一场比赛，下一集也会独立于前一集的结局。因此，所有的情节都可以被认为以相同的终点状态结束，不同的结果会有不同的奖励。这种情况的任务叫做情景任务。在情景性任务中，我们有时需要区分所有非终结状态集合 (表示 S) 和所有状态集合 (表示 S+)。终止时间 T 是一个随机变量，它通常随事件而变化。另一方面，在许多情况下，代理-环境交互并不能自然地分解为可识别的事件，而是持续不断地进行。例如，这是一种自然的方式来制定一个正在进行的过程控制任务，或者是一个长寿命机器人的应用程序。我们称之为持续任务。回归公式 (3.7) 是有问题的继续任务, 因为最后的时间步将 T =∞, 和回报, 这是我们正试图最大化, 能轻易本身

传授这种先验知识的更好的地方是最初的政策或初始价值函数，或者是对它们的影响。第 17.4 节进一步探讨了设计有效奖励信号的问题。7 集在文学作品中有时被称为“审判”。

是无限的。(例如，假设代理在每个时间步骤收到 +1 的奖励。) 因此，在这本书中，我们通常使用返回的定义，这个定义在概念上稍微复杂一点，但在数学上要简单得多。我们需要的另一个概念是折现。根据这种方法，代理尝试选择动作，以便在未来获得的折扣奖励的总和达到最大。特别是选择 At，使预期折现收益最大化:

Gt。= Rt + 1 + Rt + 2 + 2Rt + 3 + ••• = ∞ 吗?k = 0 kRt + k + 1, (3.8)

是一个参数,0 1, 称为贴现率。现值的折现率确定未来奖励: 奖励获得 k 时间步骤在未来值得 k−1 倍价值如果是立即收到。如果 < 1,(3.8) 的无限和有限值只要奖励序列 Rk 是有界的。如果 = 0, 代理“近视”是只关心最大化即时回报: 它的目标在这个案例中, 是要学会如何选择在以最大化只有 Rt + 1。如果代理的每一个行为碰巧只影响即时奖励，而不影响未来的奖励，那么近视代理可以通过分别最大化每一个即时奖励最大化 (3.8)。但总的来说，为了即时奖励最大化而采取行动，可以减少获得未来奖励的机会，从而减少回报。当 接近 1, 返回目标考虑了未来的回报更强烈; 代理人变得更有远见。连续时间的返回以一种对强化学习理论和算法很重要的方式相互关联:

Gt。= Rt + 1 + Rt + 2 + 2Rt + 3 + 3Rt + 4 + ••• = Rt + 1 +

Rt + 2 + Rt + 3 + 2Rt + 4 + •••

= Rt + 1 + Gt + 1 (3.9)

注意，这适用于所有时间步骤 t < t，即使终止发生在 t+ 1，如果我们定义 GT = 0。这通常使计算回报序列的返回变得很容易。注意, 尽管返回 (3.8) 是一个无限的术语, 它仍然是有限的, 如果非零和奖励不变 < 1。例如，如果奖励是常数 +1，那么回报是

练习 3.5 第 3.1 节中的方程是针对连续情况的，需要对其进行修改 (非常轻微) 以应用于情景性任务。显示您知道修改的内容需要提供 (3.3) 的修改版本。?

例 3.4:Pole-Balancing 这个任务的目的是运用部队手推车沿着轨道, 以防止杆铰接在车上摔倒: 据说失败发生如果过去给定杆落角垂直或如果大车运行轨道。在每次故障后，极点重新设置为垂直。这一任务可以被看作是偶然的，自然的。情节是反复尝试平衡极点。这种情况下的奖励可以是 +1，因为每次失败都没有发生，所以每次返回的次数都是失败的次数。在这种情况下，成功的永久平衡意味着无限的回归。或者，我们可以使用贴现将杆平衡视为一个持续的任务。在这种情况下, 奖励是 −1 在每个失败在其他时间和零。相关的回报在每次将 − K,K 是失败之前的时间步骤。在任何一种情况下，只要尽可能长时间保持极点平衡，回报就会最大化。

练习 3.6 假设你对待 pole-balancing 情景任务, 但也使用打折, 所有奖励 0 除了 −1 失败。那么每次的回报是多少呢? 这个回报与折现的，持续的有什么不同制定这项任务吗? ?

练习 3.7 想象你正在设计一个机器人来运行一个迷宫。你决定给它一个 +1 的奖励以奖励它从迷宫中逃跑，而在其他任何时候给它一个 0 的奖励。这个任务似乎自然而然地分成了片段——连续的贯穿整个迷宫——所以你决定把它当作一个片段式的任务，目标是最大化期望的总回报 (3.7)。在运行了一段时间的学习代理之后，您发现它在逃离迷宫方面没有任何改进。什么错了吗? 你有有效与代理商沟通你想让它实现什么? ? 运动假设 = 0.5 和 3.8 以下收到的回报序列 R1 =−1,R2 = 2,R3 = 6,R4 = 3, 和 R5 = 2,T = 5。G0 G1 G5 是什么? 提示: 向后的工作。? 锻炼 3.9 假设 = 0.9 和奖励序列 R1 = 2 无限紧随其后 7 年代序列。G1 和 G0 是什么? ? 练习 3.10 证明 (3.10) 中的第二个等式。?

3.4。情节和持续任务的统一表示法 57

3.4 片段式和连续式任务的统一表示法

在前一节中，我们描述了两种强化学习任务，一种是 agent-environment 交互自然地分解成一系列独立的片段 (情景性任务)，另一种是不存在的 (持续任务)。前一种情况在数学上更容易，因为每一个行为只影响在事件中随后获得的有限数量的奖励。在这本书中，我们有时考虑一种问题，有时考虑另一种问题，但往往同时考虑这两种问题。因此，建立一种能让我们同时准确地讨论这两种情况的符号是很有用的。要精确地描述情景任务，需要一些额外的符号。我们需要考虑一系列的事件，而不是一长串的时间步，每一个都由有限的时间步组成。我们把每一集的时间步骤从零重新开始。因此, 我们必须不仅指圣, 表示在时间 t, 但圣, 我状

态表示在时间 t 的第一集 (Rt 和类似, 我, 我, t, 我,Ti, 等等)。然而，当我们讨论情景任务时，我们几乎不需要区分不同的事件。我们几乎总是在考虑某一集的情节，或说一些对所有情节都适用的事情。因此，在实践中，我们几乎总是通过删除对情节编号的显式引用来略微滥用表示法。也就是说，我们写 St 是指 St,i，等等。我们需要另一种惯例来获得一种单一的表示法，它包括情景性和持续性的任务。我们将回报定义为一种情况下有限项的和 (3.7) 另一种情况下无限项的和 (3.8)。这两个可以通过考虑插曲结束进入一种特殊的吸收状态来统一，这种吸收状态只会转移到自身，只会产生零的奖励。例如，考虑状态转换图:

这里的实方表示一集结束时对应的特殊吸收状态。从 S0 开始，我们得到奖励序列 +1 +1 +1 +1 0 0 0 0。把这些相加，我们得到相同的回报不管是对第一个 T 奖励 (这里 T = 3) 还是对整个无穷序列。即使引入了折现，这仍然成立。因此, 我们可以定义返回, 一般来说, 根据 (3.8), 使用省略的会议集不需要数字时, 包括 = 1 的可能性, 如果和定义 (例如, 因为所有事件终止)。或者, 我们可以写

Gt。= T ?k = t + 1 t k−−1 rk, (3.11)

包括 T =∞ 的可能性或 = 1(但不是全部)。我们在本书的其余部分使用这些约定来简化符号，并表达情景任务和持续任务之间的密切相似之处。(稍后，在第 10 章，我们将介绍一种既持续又不打折扣的配方。)

58 第三章: 有限马尔可夫决策过程

3.4 策略和价值功能

几乎所有的增强学习算法都涉及到评估价值函数——状态函数 (或状态-动作对) 的函数，这些函数估计代理在给定状态下的状态有多好 (或者在给定状态下执行给定动作有多好)。这里所说的“多好”指的是未来的回报，也就是预期的回报。当然，代理人在未来可能得到的回报取决于它将采取什么行动。因此，价值函数是根据特定的行为方式 (称为策略) 来定义的。在形式上，策略是从状态到选择每个可能动作的概率的映射。如果代理政策 时间 t 后, 然后 (|) 的概率是在如果圣 =。= p, 是一个普通的函数; “|” 的 (|) 只是提醒, 它定义了一个概率分布在 (s) 为每个年代 美国强化学习方法指定代理的政策是如何改变结果的经验。练习 3.11 如果当前状态是圣, 根据随机选择和行动政策 , 那么什么是 Rt + 1 的 的期望和 four-argument 函数 p(3.2)? ? 的价值功能状态下政策 , 表示 v (s), 是预期收益后当从年代和 。mdp, 我们可以定义 v 正式通过

v (年代)。= E [Gt | 圣 = s]= E

∞ 吗?k = 0 kRt + k + 1

圣 = 年代

年代,(3.12)

在 E (•) 表示一个随机变量的期望值, 因为代理遵循政策 , 和 t 是任何时间步。注意，终端状态 (如果有的话) 的值总是 0。我们称之为政策 v 州值函数的函数。同样, 我们定义了采取行动的价值在国家年代下政策 表示 q (,), 预期收益从年代、采取的行动, 以及此后 以下政策:

q (年代)。圣 = = E [Gt | 年代, 在 =]= E

∞ 吗?k = 0 kRt + k + 1

圣 =,=

。(3.13)

我们称之为 q 政策 的行为价值函数。练习 3.12 给出方程 v q 和 。? 练习 3.13 给出方程 q v 和 four-argument p。价值函数 v 和 q 可以从经验估计。例如, 如果一个代理遵循政策 和保持平均, 每个国家遇到的实际回报之后, 状态, 然后将收敛于国家的平均价值,v (s), 遇到那种状态的次数趋于无穷。如果单独平均每个行动都在每一个州, 那么这些平均也会收敛于

动作值,q (年代)。我们称这种蒙特卡罗方法的评估方法, 因为它们涉及在许多随机样本的平均实际回报。

第五章介绍了这些方法。当然，如果有非常多的州，那么单独为每个州保持单独的平均水平可能并不实际。相反, 代理必须保持 v q 作为参数化函数 (参数少于状态) 和调整参数以更好地匹配观察到的回报。这也可以产生准确的估计，尽管这在很大程度上取决于参数化函数近似器的性质。这些可能性在本书的第二部分中进行了讨论。在增强学习和动态规划中使用的值函数的基本特性是，它们满足类似于我们已经为返回 (3.9) 建立的递归关系。任何政策和任何国家年代, 下面的一致性条件之间拥有的价值和其可能的继任者状态:

v (年代)。= E [Gt | 圣 = s] = E [Rt + 1 + Gt + 1 | 圣 = s] (通过 (3.9))

=

一个 (|)? s?

R p(s ?r | s,)

r + E (Gt + 1 | 圣 + 1 = s ?)

=

一个 (|)? r s ?, p(s ?r | s,)

r + v (?)

年代,(3.14)

其中隐含的行为，a，是从集合 a 中取的，下一个状态，s?，从集合 S 中取 (或从 S+ 中，在情景性问题中)，得到的奖励，r，是从集合 r 中取的，也就是在上一个方程中，我们如何将两个和，1 除以 S 的所有值? 另一个除以 r 的所有值，变成两个值的总和。我们经常使用这种合并和来简化公式。注意如何将最终表达式轻松地读取为预期值。它实际上是三个变量的所有值的和，a, s? 和 r。三, 我们计算概率, (|)p(s ?，r |s, a)，将括号中的数量乘以这个概率，然后求和得到期望值的所有可能性。

年代

s?

r p 一个

备份图 v 方程 (3.14) 是 v 贝尔曼方程。它表达了一个国家的价值与其继承国的价值之间的关系。设想一下，从一个国家到它可能的继承国，正如图中所建议的那样。每个开环代表一个状态，每个实环代表一个状态-动作对。从国家年代, 顶部的根节点, 代理可以采取的一些行动——三个政策 的基于图所示。从每一种情况，环境都可以对接下来的几个状态之一 s 做出反应。(图中显示了两个)，以及一个奖励，r，取决于函数 p 给出的动力学。Bellman 方程 (3.14) 的平均值除以所有的可能性，根据其发生的概率加权。它说明开始状态的值必须等于期望下一个状态的 (折现的) 值，加上沿途期望的回报。价值函数 v 是贝尔曼方程的唯一解。我们在后面的章节中展示了这个 Bellman 方程是如何形成许多方法的基础的

计算、近似和 v 学习。我们将这些图称为备份图，因为它们的关系图构成了更新或备份操作的基础，而这些操作是增强学习方法的核心。这些操作将价值信息从继承状态 (或状态操作对) 转移回状态 (或状态操作对)。我们在书中使用备份图表来提供我们讨论的算法的图形摘要。(注意，与转换图不同，备份图的状态节点不一定表示不同的状态; 例如，一个国家可能是它自己的继承者。示例 3.5:Gridworld 图 3.2(左) 显示了一个简单的有限 MDP 的矩形网格世界表示。网格的单元格对应于环境的状态。在每个单元格中，有四种操作是可能的: 北部、南部、东部和西部，这决定了代理在网格上按各自的方向移动一个单元格。行动, 将代理的网格离开它的位置不变, 但也导致奖励 −1。其他行为的奖励为 0，除了那些将 agent 从特殊状态 a 和 b 中移出的行为外，这四个行为的奖励均为 +10，并将 agent 移至 a ? 从状态 B 开始，所有行为都产生 +5 的奖励，并将代理带到 B?

图 3.2:Gridworld 示例: 等可能随机策略的异常奖励动态 (左) 和状态值函数 (右)。

假设代理在所有状态中选择所有四个具有相同概率的操作。图 3.2(右) 显示了值函数,v , 对于这一政策, = 0.9 折扣奖励情况。这个值函数是通过求解线性方程组 (3.14) 得到的。注意下边缘附近的负值; 这是在随机策略下撞到网格边缘的高概率的结果。状态 A 是这个策略下

的最佳状态，但是它的预期回报小于 10，它的即时回报，因为从代理被带到 A?，它很可能从那里进入网格的边缘。另一方面，状态 B 的值大于 5，它的即时奖励，因为从 B 代理被带到 B?，它有一个正值。从 B 吗? 可能撞到边的预期惩罚 (负奖励) 比可能撞到 A 或 B 的预期收益要多。练习 3.14 贝尔曼方程 (3.14) 为每个国家的价值函数必须持有 v 如图 3.5 3.2(右) 的例子。显示数值状态方程适用于中心, 价值 + 0.7, 关于它的四个邻国, 价值 + 2.3 + 0.4−0.4,+ 0.7。(这些数字只精确到小数点后一位。) 在 gridworld 示例中，奖励对于目标是积极的，对于跑到世界的边缘是消极的，其余时间则为零。是这些迹象吗

奖励是重要的，还是仅仅是他们之间的间隔? 使用 (3.8) 证明，在所有奖励中增加一个常数 c，会给所有状态的值增加一个常数 vc，因此不会影响任何策略下任何状态的相对值。什么是 vc c 和 吗? ? 练习 3.16 现在考虑在情景性任务 (如迷宫跑步) 的所有奖励中增加一个常数 c。这是否会产生任何影响，或者它会不会像上述的持续任务一样，将任务保持不变? 为什么或为什么不呢? 给一个例子。吗? 例 3.6: 高尔夫制定打一个洞的高尔夫作为强化学习任务, 我们计算一个点球 (负奖励)−1 为每个中风直到我们击球进洞里。状态是球的位置。状态的值是从那个位置到洞的笔画数的负数。我们的行动就是我们如何瞄准球，如何挥拍球，当然还有我们选择哪个球杆。让我们以前者为例，考虑俱乐部的选择，我们假设它是一个推杆还是一个车手。图 3.3 的上半部分显示了一个可能的状态值函数 vputt(s)。

Q *(年代, 驱动程序) V 推杆沙子

绿色的年代 n 维

沙子

绿色是个 n 维 vputt

问 ∗(年代, 司机)

图 3.3: 一个高尔夫示例: 状态值 func- 设置 (上) 和使用驱动 (下) 的最优动作值函数。总是使用推杆。洞中的终端状态的值为 0。从绿色的任何地方我们假设我们可以做一个推杆; 这些国家有价值 −1。在绿色的地方，我们不能通过放置来到达洞，而且价值更大。如果我们能从国家通过将达到绿色, 那一个国家必须有价值不到绿色的价值, 也就是说,−2。为了简单起见，让我们假设我们可以非常精确和确定地推推，但范围有限。这给了我们锋利的轮廓线标记 −2 图中; 在这条线和绿线之间的所有位置都需要精确地划两笔才能完成这个洞。同样, 把范围内任何位置 −2−3 轮廓线必须有一个价值, 等等, 在图中所示的轮廓线。让不让我们摆脱砂陷阱, 所以他们有价值的 −∞。总的来说，我们要用 6 次击球才能从发球台到球洞。

R s? 年代,

一个? p

q 备份图 3.17 运动的贝尔曼方程是什么动作值, 也就是说,q 吗? 它必须给行动价值 q (,) 的动作值,q (s ? 提示: 右边的备份图对应于这个方程。显示类似于 (3.14) 的方程序列，但用于操作值。?

62 年第三章: 有限马尔可夫决策过程

一个国家的价值取决于该国家可能采取的行动的价值，以及在当前政策下采取每一个行动的可能性。我们可以把这看作是建立在状态的一个小备份图，并考虑每个可能的行动:

给相对应的方程这直觉和图根节点的值,v (s) 的价值预期的叶子节点,q (,), 鉴于圣 = s。这个方程应该包括一个期望条件政策后, 。然后给第二个方程的期望值是写出明确的 (|) 这样方程中就不会出现期望值符号。? 练习 3.19 一个行动的价值,q (,), 取决于预期的未来回报和预期的和剩下的奖励。再一次，我们可以把它想象成一个小的备份图，这个以动作 (状态-动作对) 为基础，分支到可能的下一个状态:

给对应的方程为行动的价值, 这个直觉和图 q (,), 预计下一个奖励,Rt + 1, 和预期的下一个状态值,v (圣 + 1), 考虑到圣 = 和 = 一个。这个等式应该包含一个期望，而不是一个条件下遵循政策。然后给出第二个方程，用 p(s) 显式地写出期望值。，r |s, a) 由 (3.2) 定义，不存在期望值符号出现在等式中。?

3.5 最优策略和最优值函数

解决强化学习任务大致是指找到一种策略，从长期来看能获得很多回报。对于有限的 MDPs，我们可以用以下方法精确地定义一个最优策略。值函数定义策略的部分排序。策略定义 是优于或等于 政策呢? 如果它的预期收益大于或等于 的? 所有国家。换句话说， 吗? 当且仅当 v (s) v ?(s) 为所有年代 美国总有至少一个政策优于或等于其他政策。这是一个最优的策略。虽然可能会有不止一个, 我们都表示最优政策 ∗。他们共享相同的状态值函数, 称为最优状态值函数, 表示 v∗, 定义为 v∗(年代)。= 最大 v (s), (3.15) 年代。

最优政策也共享相同的最优行为价值函数, 表示为 q∗, 定义为

问 ∗(年代)。= 最大 q (,), (3.16) 对于所有 年代和 (s)。对于状态-动作对 (s, a)，该函数给出在状态 s 中采取动作 a 的期望回报，并在之后遵循最优策略。因此, 我们可以写问 ∗∗v 如下: 问 ∗(,)= E(Rt + 1 + v∗(圣 + 1)| 圣 =,=)。(3.17)

例 3.7: 最优值函数为高尔夫球的下方图 3.3 显示了一个可能的最优行为价值的轮廓函数 q∗(年代, 司机)。这些是每个状态的值，如果我们先和司机一起玩一次，然后选择司机或推杆，以哪个更好。驾驶员使我们能把球打得更远，但精确度较低。只有当我们离得很近的时候，我们才能用驾驶员一枪就能到达那个洞; 从而为问 ∗−1 轮廓 (年代, 司机) 只涉及一小部分的绿色。然而, 如果我们有两个中风, 然后我们可以从多远, 到达洞 −2 所示的轮廓。在这种情况下我们不需要开车到小 −1 内轮廓, 但只有在绿色; 从那里我们可以使用推杆。最优动作-值函数在将一个特定的第一个动作提交给驱动程序之后给出值，在本例中是这样的，但是在之后使用任何最好的动作。−3 轮廓仍远, 包括起始三通。从发球台开始，最好的动作顺序是两个发球点和一个推杆，三次击球使球下沉。因为 v∗ 的价值函数是一个政策, 它必须满足的贝尔曼方程的自洽性条件状态值 (3.14)。然而, 因为它是最优值函数 v∗ 的一致性条件可以写在一个特殊形式没有提及任何具体的政策。这是贝尔曼方程 v∗ 或贝尔曼最优方程。直觉上，Bellman 最优方程表达了这样一个事实，即最优策略下的状态的值必须等于该状态的最佳行为的期望回报:

v∗(s)= max (s)q ∗(,) = 最大一个 E ∗(Gt | 圣 = s =)
= 最大一个 E ∗(Rt + 1 + Gt + 1 | 圣 =,=) (通过 (3.9))
= 最大一个 E(Rt + 1 + v∗(圣 + 1)| 圣 =,=) (3.18)
= 最大一个
r s ?, p(s ?r | s,) r + v∗(s ?)?。(3.19)

最后两个方程两种形式的 v∗ 的贝尔曼最优方程。问 ∗ 的贝尔曼最优方程

问 ∗(,)= E
Rt + 1 + max 一个? 问 ∗(圣 + 1 ?) 吗? 吗? 吗?St = s,At = a。
=
r s ?, p(s ?r | s,)
r + max 一个? 问 ∗(s ?, 一个?)
。(3.20)

下图中的备份图显示图形的跨越未来状态和行为被认为是在贝尔曼最优性方程 v∗∗。这些都是一样的备份 v 图和 q 除了弧添加了早些时候在代理的选择指向代表最大的选择是采取而不是期望值给出一些政策。左边的备份图用图形表示 Bellman 最优方程 (3.19)，右边的备份图用图形表示 (3.20)。

图 3.4: 备份为 q v∗∗ 图

v 有限 mdp, 贝尔曼最优方程 (3.19) 有一个独特的解决方案独立的政策。Bellman 最优方程实际上是一个方程组，每个状态一个，如果有 n 个状态，那么 n 个未知数中有 n 个方程。如果环境的动力学 p 是已知的, 那么原则上可以解决这个方程组 v∗ 使用任何一个各种各样的方法求解非线性方程组。一个人可以问 ∗ 解一组相关的方程。一旦一个 v∗, 它相对容易确定最优政策。对于每个状态 s，在 Bellman 最优方程中会有一个或多个动作来获得最大值。任何只给这些行为分配非零概率的策略都是最优策略。你可以把这看作是一步搜索。如果你有

最优值函数,v∗, 之后出现的行为最好的一步法搜索最优行为。另一种说法是, 任何政策都是贪婪与最优评价函数 v∗ 是最优策略。“贪婪”一词在计算机科学中被用来描述任何一种搜索或决策程序，它只基于本地或即时的考虑，而不考虑这样的选择可能会阻止未来获得更好的选择。因此，它描述了只根据短期后果选择行动的政策。v∗ 的美妙之处在于, 如果一个用它来评估 actions-specifically 的短期后果, 一步后果贪婪策略实际上是最佳的长期意义上我们感兴趣, 因为 v∗ 已经考虑到所有可能的未来的回报后果的行为。通过 v∗, 最优预期长期回报是本地和变成一个量立即对每个状态。因此，一步一步的搜索会产生长期的最佳行动。有问 ∗ 使选择最佳的操作更简单。问 ∗, 甚至代理不需要做一个领先一步搜索: 对于任何一个国家, 它可以找到任何行动最大化问 ∗(年代)。行为价值函数有效地缓存所有领先一步的搜索结果。它提供了最佳的预期长期回报，作为每个状态-动作对的局部和立即可用的值。因此，代价是。

表示状态-动作对的函数，而不仅仅是状态，最优动作-值函数允许选择最优动作，而不需要知道任何可能的后续状态及其值，也就是说，不需要知道环境的动态。例 3.8: 解决 Gridworld 假设我们解决贝尔曼方程 v∗ 为简单的网格任务在例 3.5 中引入的, 在图 3.5(左)。回想一下，状态 A 后面跟着 +10 的奖励，然后过渡到状态 A?，而状态 B 之后是 +5 的奖励，然后过渡到状态 B? 图 3.5(中间) 显示了最优值函数，图 3.5(右) 显示了相应的最优策略。在单元格中有多个箭头的地方，所有相应的操作都是最优的。

22.0 24.4 22.0 19.4 17.5。

19.8 22.0 19.8 17.8 16.0

17.8 19.8 17.8 16.0 14.4

14.0 14.0 14.0 14.0 14.0 14.0 14.0 14.0 14.0 14.4 13.0 11.7 一个 B

一个“B + 10 + 5

)gridworld b)V * c)* gridworld V∗ ∗ 图 3.5:gridworld 示例的最佳解决方案。

例 3.9: 回收机器人的 Bellman 最优方程 (3.19)，我们可以显式地给出回收机器人的 Bellman 最优方程。为了使事情更紧凑，我们将状态分为高状态和低状态，分别用 h、l、s、w 和 re 进行动作搜索、等待和充值。因为只有两个状态，Bellman 最优方程由两个方程组成。v∗ 方程 (h) 可以写成:

v∗(h)= max

p(h | h,s)[r(h,s、h)+ v∗(h)]+ p(l | h,s)[r(h,s、l)+ v∗(l)],p(h | h,w)[r(w h,h)+ v∗(h)]+ p(l | h,w)[r(h、w、l)+ v∗(l))

= 最大

(rs + v∗(h))+(1−)[rs + v∗(l)],1[rw + v∗(h)]+ 0(rw + v∗(l))

= 最大

rs + (v∗(h)+(1−)v∗(l)],rw + v∗(h)

。

遵循同样的步骤为 v∗(l) 收益率方程

v∗(l)= max rs−3(1−)+ [(1−)v∗(h)+ v∗(l)],rw + v∗(左), v∗(h) 。对于任何选择 rs,rw, , 和 0 < 1 0 , 1, 正是一对数字,v∗∗(h) 和 v(l), 同时满足这两个非线性方程组。

显式求解贝尔曼最优方程为求解最优策略提供了一条途径，从而解决了强化学习问题。然而，这种解决方案很少直接有用。它类似于一个彻底的搜索，展望所有的可能性，计算它们发生的概率和他们的期望回报。这个解决方案依赖于至少三个在实践中很少出现的假设:(1) 我们准确地知道环境的动态;(2) 我们有足够的计算资源来完成解的计算;(3) 马尔可夫性质。对于我们感兴趣的任务类型，一个人通常无法实现这个解决方案，因为这些假设的各种组合都被违反了。例如，虽然第一个和第三个假设对西洋双陆棋没有问题，但是第二个假设是主要的障碍。因为游戏大约有 1020 个国家, 它将会花上几千年今天最快的计算机来解决 v∗ 贝尔曼方程, 和寻找问 ∗ 也是如此。在强化学习中，人们通常不得不满足于近似解。许多不同的决策方法可以看作是近似求解 Bellman 最优方程的方法。例如, 启发式搜索方法可以被视为扩大的右边 (3.19) 几次, 一些深度, 形成一个“树”的可能性, 然后使用启发式评估函数近似 v∗“叶子”节点。(启发式搜索方法如 ∗ 几乎总是基于情景的情况。) 动态规划方法与 Bellman

最优方程的关系更为密切。许多强化学习方法可以被清楚地理解为近似求解 Bellman 最优方程，使用实际的经验转换来代替预期转换的知识。我们将在下一章中考虑各种这样的方法。

练习 3.20 绘制或描述高尔夫例子的最优状态值函数。吗? 练习 3.21 绘制或描述最优动作-值函数的轮廓将 q∗(年代, 推杆), 高尔夫的例子。?

+ 2 0 0 + 1 左右练习 3.22 考虑右边显示的持续的 MDP。唯一要做的决定是在最上面的状态中，有两个动作可用，左和右。这些数字显示了每次行动后所获得的决定性奖励。正好有两个决定性的政策, left right。最优如果 = 0 是什么政策? 如果 = 0.9 ? 如果 = 0.5 ? ? 练习 3.23 给贝尔曼方程问 ∗ 回收机器人。吗? 图 3.5 给出网格世界的最佳状态的最佳值为 24.4，到小数点后一位。使用你的最优策略知识和 (3.8) 来象征性地表达这个值，然后将其计算到小数点后三位。?

练习 3.25 v 的给一个方程的问 ∗∗。?

练习 3.26 给出方程问 v 的 ∗∗four-argument p。?

3.7。最优性和近似 67 年

练习 3.27 给出方程 ∗ 问 ∗。? 练习 3.28 给出方程 ∗∗v 和 four-argument p。练习 3.29 改写四个贝尔曼方程四个值函数 (v ,v∗、q 和 q∗) 的三个参数函数 p(3.4) 和双参数函数 r (3.5)。?

3.6 最优性和近似

我们定义了最优值函数和最优策略。显然，了解最优策略的代理做得非常好，但在实践中这种情况很少发生。对于我们感兴趣的任务类型，最优策略只能以极高的计算成本生成。定义良好的最优性概念组织了我们在本书中描述的学习方法，并提供了一种理解各种学习算法的理论性质的方法，但理想情况是，代理只能在不同程度上近似。正如我们上面所讨论的，即使我们有一个完整而准确的环境动力学模型，通常也不可能仅仅通过求解 Bellman 最优方程来计算最优策略。例如，像国际象棋这样的棋类游戏只是人类经验的一小部分，然而大型定制设计的计算机仍然无法计算出最佳的移动速度。代理所面临的问题的一个关键方面是，它总是可用的计算能力，特别是它可以在单个时间步中执行的计算量。可用内存也是一个重要的限制。通常需要大量内存来构建价值函数、策略和模型的近似。在具有小的、有限状态集的任务中，可以使用数组或表来形成这些近似，每个状态 (或状态操作对) 有一个条目。我们把这个叫做表格式案例，以及我们称之为表格法的相应方法。然而，在许多实际感兴趣的情况下，有比表中可能的条目多得多的状态。在这些情况下，必须使用某种更紧凑的参数化函数表示来逼近函数。强化学习问题的框架迫使我们接受近似。然而，它也为我们提供了一些实现有用近似的独特机会。例如，在近似最优行为时，代理可能面临许多状态，其概率非常低，因此为其选择次优行为对代理收到的报酬影响不大。例如，泰索罗的西洋双陆棋棋手，即使在与专家的比赛中，他可能会在棋局配置上做出非常糟糕的决定。事实上,TD-Gammon 可能使错误决策的一大部分游戏的状态集。在线强化学习的性质使得它可以近似最优政策的方式把更多的精力投入到学习为常见状态做出正确的决定, 以牺牲较少的努力很少遇到。这是区分强化学习和近似求解 MDPs 的其他方法的一个关键属性。

3.7 总结

让我们总结一下我们在本章中提出的强化学习问题的要素。强化学习是指从互动中学习如何行为以达到目标。增强学习代理及其环境通过一系列离散时间步骤进行交互。它们的接口规范定义了一个特定的任务: 操作是代理所做的选择; 国家是作出选择的基础; 奖励是评估选择的基础。代理商内部的一切都被代理商完全了解和控制; 外界的一切都是不完全可控的，但可能是也可能不是完全可知。策略是一种随机规则，其中代理选择作为状态的函数。代理人的目标是在一段时间内获得最大的回报。当前面描述的强化学习设置用定义良好的转移概率

表示时，它构成了一个马尔可夫决策过程 (MDP)。一个有限的 MDP 是一个具有有限状态、作用和 (正如我们在这里所描述的) 奖励集的 MDP。目前的强化学习理论大多局限于有限的 MDPs，但方法和思想的应用更为普遍。回报是 agent 寻求最大化 (期望值) 的未来回报的函数。它有几个不同的定义，这取决于任务的性质，以及人们是否希望降低延迟奖励。不打折扣的提法适用于情景任务，在情景任务中，参与者与环境的交互自然地分解为多个片段; 折后的公式适用于持续的任务，在这些任务中，交互不会自然地分解为多个片段，而是无限地继续。我们试图定义这两种任务的回报比如一组方程可以同时适用于情景和连续的情况。如果代理使用策略，则策略的值函数分配给每个状态或状态操作对、该状态的预期回报或状态操作对。分配给每个状态或状态操作对的最优值函数，是任何策略所能实现的最大期望回报。价值函数是最优的策略是最优策略。对于给定的 MDP，状态和状态操作对的最优值函数是唯一的，但是有许多最优策略。任何对最优值函数贪婪的策略都必须是最优策略。Bellman 最优方程是最优值函数必须满足的特殊一致性条件，原则上可以求解最优值函数，从而相对容易地确定最优策略。强化学习问题可以以各种不同的方式提出，这取决于对最初提供给代理的知识水平的假设。在完全知识的问题上，agent 对环境的动力学有一个完整而准确的模型。如果环境是 MDP，则该模型由完整的四参数动态函数 p(3.2) 组成。在知识不完整的问题上，没有一个完整的、完美的环境模型。即使代理拥有完整和准确的环境模型，代理通常也无法执行足够的每次步骤计算来充分使用它。可用内存也是一个重要的限制。构建可能需要内存

精确逼近值函数、策略和模型。在大多数实际感兴趣的情况下，表中的状态远远多于可能的条目，必须进行近似。定义良好的最优性概念组织了我们在本书中描述的学习方法，并提供了一种理解各种学习算法的理论性质的方法，但强化学习代理只能在不同程度上近似是理想的。在强化学习中，我们非常关注那些不能找到最优解但必须以某种方式近似的情况。

强化学习问题是最优控制领域的马尔可夫决策过程思想的产物。这些历史影响和其他来自心理学的主要影响在第一章的简要历史中被描述。强化学习增加了 MDPs 对逼近和不完全信息的关注，以解决实际的大问题。MDPs 和强化学习问题与传统的人工智能学习和决策问题仅有微弱的联系。然而，人工智能现在正从不同的角度积极探索 MDP 的规划和决策公式。MDPs 比以前在人工智能中使用的公式更普遍，因为它们允许更一般的目标和不确定性。例如，Bertsekas (2005)，White (1969)，Whittle(1982, 1983) 和 Puterman(1994) 研究了 MDPs 的理论。Ross(1983) 对有限情况做了一个特别紧凑的处理。在随机最优控制的标题下也研究了多目标多目标规划，其中自适应最优控制方法与强化学习的关系最为密切 (如 Kumar, 1985;Kumar 和 Varaiya,1986)。多边开发计划署的理论是从努力理解在不确定的情况下做出一系列决策的问题演变而来的，在这种情况下，每个决策都可以依赖于以前的决策及其结果。它有时被称为多级决策过程理论, 或顺序决策过程, 在序贯抽样统计文学的根开始论文的汤普森 (1933、1934) 和罗宾斯 (1952), 我们将在第二章与土匪问题 (典型的 mdp 如果制定 multiple-situation 问题)。使用 MDP 形式主义讨论强化学习的最早实例是 Andreae (1969b) 对学习机器的统一观点的描述。Witten and Corbin(1973) 利用 MDP 形式主义实验了后来 Witten (1977, 1976a) 分析的强化学习系统。尽管 Werbos(1977) 没有明确地提到 MDPs，但他对与现代强化学习方法相关的随机最优控制问题提出了近似的解决方法 (参见 Werbos, 1982, 1987, 1988, 1989, 1992)。尽管 Werbos 的想法在当时并没有得到广泛的认可，但他们有先见之明，强调了在包括人工智能在内的各种领域中近似解决最优控制问题的重要性。强化学习和 MDPs 最具影响的整合是由于 Watkins(1989)。

3.1 我们用 p(s?r |s a) 是有点不寻常。在 MDP 文献中，用状态跃迁概率 p(s? 在强化学习中，我们更多的是参考个体实际的或样本的奖励 (而不仅仅是他们的期望值)。我们的表示法也清楚地表明 St 和 Rt 通常是共同确定的，因此必须具有相同的时间索引。在强化学习教学中，我们发现我们的表示法在概念上更直观，更容易理解。要更好地直观地讨论系统理论的状态概念，请参阅明斯基 (1967)。生物反应器的例子基于 Ungar(1990) 和 Miller 和 Williams(1992) 的成果。回收机器人的例子是受到乔纳森 • 康奈尔 (Jonathan Connell)(1989) 发明的 can-collect 机器人的启发。Kober 和 Peters(2012) 展示了增强学习的机器人应用。

3.2 奖赏假说由 Michael Littman 提出 (personal communica-) 变形)。3.3-4 情景任务和持

续任务的术语与通常不同用于 MDP 文献。在该文献中，通常区分三种类型的任务:(1) 有限范围任务，其中交互在特定的固定时间步数之后终止;(2) 不确定视界任务，其中交互可以任意长时间，但最终必须终止; 和 (3) 无限视界任务，其中相互作用不终止。我们的情景任务和连续任务分别类似于不确定的视界和无限的视界任务，但我们更喜欢强调交互性质的不同。这种差异似乎比通常术语所强调的目标函数的差异更为根本。情景性任务通常使用不确定层目标函数，而连续任务则使用无限层目标函数，但我们认为这是一个共同的巧合，而不是根本的区别。平衡杆的例子来自于 Michie 和 Chambers(1968)、Barto、Sutton 和 Anderson(1983)。3.5-6 根据长期的好坏来分配价值由来已久的根源。在控制理论中，将状态映射到代表控制决策的长期后果的数值是最优控制理论的关键部分，该理论是在 20 世纪 50 年代通过扩展 19 世纪经典力学的状态函数理论而发展起来的 (参见，Schultz 和 Melsa, 1967)。Shannon(1950) 在描述计算机如何被编程去玩国际象棋时，建议使用一个考虑到国际象棋位置的长期优势和劣势的评估函数。沃特金斯 (1989) 的 q 学习算法估计问 $*$(第六章) 行为价值功能强化学习的一个重要组成部分, 因此

这些函数通常被称为“q 函数”。但动作价值函数的概念要比这个古老得多。Shannon(1950) 认为，一个国际象棋程序可以使用一个函数 h(P,M) 来决定移动 M 是否值得探索。Michie’s(1961, 1963) 的威胁系统和 Michie 和 Chambers(1968) 的盒子系统可以理解为估算行动价值函数。在经典物理学中，哈密尔顿的主函数是一个动作-值函数; 牛顿动力学对于这个函数是贪婪的 (例如，Goldstein, 1957)。动作-值函数在 Denardo(1967) 关于收缩映射的动态规划的理论处理中也发挥了中心作用。贝尔曼最优方程 ($v*$) 推广由理查德 • 贝尔曼 (1957 年), 他称之为 “基本函数方程。“连续时间和状态问题的 Bellman 最优方程的对应关系被称为哈密顿-雅可比- Bellman 方程 (或者通常是哈密顿-雅可比-雅可比方程)，表明它起源于古典物理学 (例如，Schultz 和 Melsa, 1967)。高尔夫的例子是由克里斯 • 沃特金斯提出的。

4. 第四章动态规划

术语动态规划 *(DP)* 指的是一组算法，它们可以用来计算最优策略，给出了一个完美的环境模型，作为 *Markov* 决策过程 *(MDP)*。传统的 *DP* 算法在强化学习中的效用是有限的，这一方面是因为它们假设了一个完美的模型，另一方面是因为它们的计算开销很大，但它们在理论上仍然很重要。*DP* 为理解本书其余部分的方法提供了基本的基础。事实上，所有这些方法都可以被看作是试图达到与 *DP* 几乎相同的效果，只需要较少的计算和不假设环境的完美模型。我们通常假设环境是一个有限的 *MDP*。也就是说，我们假设它的状态，作用和奖赏集 *S A R* 是有限的，它的动力学是由一组概率 *p(S ?,r | s)* 年代, *(s),r r* 和 *s ? S +(S + S +* 终端状态如果问题情景*)*。尽管 *DP* 思想可以应用于具有连续状态和操作空间的问题，但只有在特殊情况下才可能有精确的解决方案。获得具有连续状态和操作的任务的近似解的一种常见方法是量化状态和操作空间，然后应用有限状态 *DP* 方法。我们在第 *9* 章中探讨的方法适用于连续问题，是该方法的重要扩展。*DP* 和强化学习的关键思想是利用价值函数来组织和组织寻找好的策略。在本章中，我们将展示如何使用 *DP* 来计算第 *3* 章定义的值函数。如前所述, 我们可以很容易地获得最优政策一旦我们找到了最优值函数,*v∗* 或者 *q∗*, 满足贝尔曼最优方程*: v∗(s)= max* 一个 *E(Rt + 1 + v∗(*圣 *+ 1)|* 圣 *=,=)*

= 最大一个

r s ?, p(s ?r | s,)

r + v∗(?)

, 或 *(4.1)*

问 *∗(,)= E*

Rt + 1 + max 一个*?* 问 *∗(*圣 *+ 1 ?)* 吗*?* 吗*?* 吗*?* 圣 *=,=*

=

r s ?, p(s ?r | s,)

r + max 一个*?* 问 *∗(s ?,* 一个*?)*

, (4.2)

年代, *(s),* 和 *s ? S +*。正如我们将看到的，*DP* 算法是通过将诸如此类的 *Bellman* 方程转换为赋值，也就是说，转换为更新规则以改进期望值函数的近似。

4.1 政策评估 (预测)

首先, 我们考虑如何计算一个任意的州值函数 v 政策 。这在 DP 文献中被称为政策评估。我们也把它称为预测问题。从第三章, 年代,

v (年代)。= E [Gt | 圣 = s] = E [Rt + 1 + Gt + 1 | 圣 = s] (从 (3.9)) = E [Rt + 1 + v (圣 + 1)| 圣 = s] (4.3)

=
一个 (|)? r s ?, p(s ?r | s,)
r + v (?)
, (4.4)

在 (|) 的概率是采取行动在国家年代政策 , 和期望是下标在 来表示他们是有条件的被跟踪。的存在唯一性 v 保证只要 < 1 或最终终止与所有国家在政策保证 。如果环境的动力是完全已知的, 那么 (4.4) 是一个系统的 | | | 年代线性方程组 | 未知 (的 v (S) 年代), 原则上, 它的解决方案是一种简单, 如果繁琐, 计算。对于我们的目的，迭代解决方法是最合适的。考虑一个近似函数序列 v0, v1, v2⋯，每个映射 S+ 到 R(实数)。初始近似,v0, 任意选择 (除了终端状态, 如果有的话, 必须给定值 0), 并且每个逐次逼近获得通过使用 v 贝尔曼方程 (4.4) 作为一个更新规则:

vk + 1(s)。= E [Rt + 1 + vk(圣 + 1)| 圣 = s] =
一个 (|)? r s ?, p(s ?r | s,)
r + vk(?)
, (4.5) 所有年代 s . 显然,vk = v 定点这个更新规则, 因为贝尔曼方程 v 保证我们平等的在这种情况下。事实上, 序列 vk 可以显示一般收敛于 v k→∞ 在相同条件下保证 v 的存在。这种算法称为迭代策略评估。生产每一个逐次逼近, 从 vk vk + 1, 迭代政策评估相同的操作适用于每个州史: 它取代年代的旧值与一个新值从旧值获得的继任者的年代, 和预期的即时回报, 以及所有可能一步过渡政策的评估。我们将这种操作称为预期更新。迭代策略评估的每次迭代更新每个状态的值一次，生成新的近似值函数

vk + 1。有几种不同类型的预期更新，这取决于一个状态 (如这里) 或状态-操作对是否正在更新，并且取决于后续状态的估计值的精确方式。DP 算法中完成的所有更新都被称为预期更新，因为它们基于对所有可能的下一个状态的期望，而不是基于一个示例下一个状态。更新的性质可以用公式表示，如上面所示，也可以用备份图表示，如第 3 章所介绍的。例如，与迭代策略评估中使用的预期更新相对应的备份图表显示在第 59 页。要编写一个连续的计算机程序来实现 (4.5) 所给出的迭代策略评估，您必须使用两个数组，一个用于旧值，一个用于新值，一个用于新的值，vk+1(s)。使用两个数组，可以从旧值中逐个计算新值，而不需要更改旧值。当然，更容易使用一个数组并更新值“到位”，也就是说，每个新值立即覆盖旧值。然后，根据状态更新的顺序，有时在 (4.5) 的右边使用新值而不是旧值。这也就地算法收敛于 v ; 实际上，它的收敛速度通常比双数组版本要快，正如您所期望的那样，因为它在新数据可用时就会使用它们。我们认为更新是在状态空间中执行的。对于就地算法，状态值在扫描过程中更新的顺序对收敛速度有显著影响。当我们想到 DP 算法时，我们通常会想到合适的版本。在下面的框中，一个完整的就地版本的迭代策略评估显示在伪代码中。注意它是如何处理终止的。在形式上，迭代策略评估只在极限上收敛，但在实践中，它必须在极限下停止。伪代码测试数量马克斯 年代 | vk + 1(S)−vk(S)| 每次扫描和停止时足够小。

示例 4.1 考虑如下所示的 4×4 gridworld。

r = Rt 1 1 −1 8 9 10 11 行动 12 13 14

非终端状态为 S =1,2，⋯14 。在每个状态中都可能有 4 个动作，A = up, down, right, left，它们决定导致相应的状态转换，除了那些将代理从网格中移除的动作实际上保持状态不变。因此, 例如,p(6−1 | 5 右)= 1,p(7−1 | 7 右)= 1,p(10 r | 5 右)= 0 r r . 这是一个尚未完全, 情景任务。奖励是 −1 在所有转换到终端状态。终端状态在图中被阴影化 (虽然它在两个地方显示，但它在形式上是一个状态)。因此预期回报函数 r(s,s)=−1 对所有国家 s,s ？假设代理遵循等概率随机策略 (所有行为的概率相等)。图 4.1 的左侧显示了由迭代策略评估计算的值函数 vk 的序列。最后估计实际上是 v , 在这种情况下, 给每个州的否定预期数量的步骤的状态, 直到终止。练习 4.1 在例 4.1 中, 如果 是等概率的随机策略, 什么是 q (11 日)? 什么是 q (7)? ？在示例 4.1 中，假设一个新的状态 15 被添加到状态 13 下面的 gridworld 中，它的操作，左，右，右，向下，分别将代理发送到状态 12、13、14 和 15。假设来自原始状态的转换没有改变。那么, 什么是 v (15) 等概率的随机的政策? 现在假设状态 13 的动力学也发生了

变化，比如从状态 13 向下的动作将代理带到新的状态 15。什么是 v (15) 为等概率的随机政策在这种情况下?? 4.3 运动的方程是什么类似于 (4.3),(4.4),(4.5) 和行为价值函数 q 及其函数序列 q0 逐次逼近,q1、q2,。。。?

4.2 政策改进

我们计算策略的值函数的原因是为了帮助找到更好的策略。假设我们有确定的值函数 v 任意确定的政策。对于一些国家年代我们想知道我们是否应该改变政策以确定性选择一个行动? = (年代)。我们都知道它是多么好未嫁的现行政策是 v (s), 但会更好或更糟改变到新的政策? 回答这个问题的一种方法是考虑在 s 和之后选择 a

Vk 的随机策略 f r t 该政策 v

-1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 k = 1 0.0

-1.7 -2.0 -2.0 -1.7 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -1.7 -2.0 -2.0 -1.7 k = 2 0.0

0.0

-2.4 -2.9 -3.0 -2.4 -2.9 -3.0 -2.9 -2.9 -3.0 -2.9 -2.4 -3.0 -2.9 -2.4 k = 3 0.0

0.0

-6.1 -8.4 -9.0 -6.1 -7.7 -8.4 -8.4 -8.4 -8.4 -7.7 -6.1 -9.0 -8.4 -6.1 k = 10 0.0

0.0

-14 年。-20 年。-22 年。-14 年。-18 年。-20 年。-20 年。-20 年。-20 年。-18 年。-14 年。-22 年。-20 年。-14 年。k = 0.0

0.0 77 年

贪心政策 g p 李。v

随机的政策

图 4.1: 小网格环境下迭代策略评估的收敛性。左列随机策略的状态-值函数的近似序列 (所有操作的可能性相等)。右列是与值函数估计值相对应的贪婪策略序列 (箭头表示达到最大值的所有操作，而数字则四舍五入为两位数)。最后的政策只保证是一个对随机策略的改进，但是在这种情况下，以及第三次迭代之后的所有策略，是最优的。

现有政策后, 。这种行为方式的价值是。

q (年代)。= E[Rt + 1 + v (圣 + 1)| 圣 =,=) (4.6)

=

r s ?, p(s ?r | s,)

r + v (?)

. 电

标准的关键是是否大于或小于 v (年代)。如果是大的, 最好是选择一个在年代, 此后跟随比它会遵循 时间一个期望它会更好的选择每次遇到年代, 事实上, 新政策将是一个更好的一个整体。这是一个被称为政策改进定理的一般结果的特例。让 ? 是任何对确定性的政策, 对于所有 年代,

q (年代, ?(s)) v (年代)。(4.7) 那么政策 ? 必须一样好, 或者比, 。也就是说, 它必须获得大于或等于预期收益从所有国家 年代:

v ?(s) v (年代)。(4.8) 而且，如果在任何状态下都有严格的 (4.7) 不等式，那么在那个状态下一定有严格的 (4.8) 不等式。这个结果尤其适用于这两个政策, 我们认为在前款规定的, 最初确定的政策, , 以及政策的改变, 吗? 是相同的, 除了 ?(s)= a ? = (年代)。显然,(4.7) 在以外的所有国家。因此, 如果 q (,)> v (s), 那么改变政策确实比 。政策改进定理证明背后的思想很容易理解。从 (4.7), 我们不断扩大 q 端 (4.6) 和 (4.7), 直到我们上重新得到 v ?(s):

v (s) q (年代, ?(s)) = E[Rt + 1 + v (圣 + 1)| 圣 =,= ?(s)] (通过 (4.6)) = E ?(Rt + 1 + v (圣 + 1)| 圣 = s] E 吗?(Rt + 1 + q (圣 + 1, ?(圣 + 1)| 圣 = s] (通过 (4.7)) = E ?(Rt + 1 + E ?(Rt + 2 + v (圣 + 2)| 圣 + 1 + 1 = ?(圣 + 1)]| 圣 = s]= E 吗?

Rt + 1 + Rt + 2 + 2v (圣 + 2) 吗? 吗? 圣 = 年代
E 吗?
Rt + 1 + Rt + 2 + 2Rt + 3 + 3v (圣 + 3)? 吗? 圣 = s ? . 电
E 吗?
Rt + 1 + Rt + 2 + 2Rt + 3 + 3Rt + 4 + • • • 吗? 吗? 圣 = 年代
= v ?(年代)。到目前为止，我们已经看到，在给定策略及其价值函数的情况下，如何轻松地评估单个状态下的策略更改为特定的操作。这是一个自然的延伸

考虑改变在所有国家和所有可能的行动, 选择在每个州的行动似乎最好根据 q (年代)。换句话说, 考虑新的贪婪的政策, 吗?, 由

?(s)。= argmax 一个 q (年代)
= argmax 一个 E(Rt + 1 + v (圣 + 1)| 圣 =,=) (4.9)
= argmax 一个
r s ?, p(s ?r | s,)
r + v (?)

其中 argmaxa 表示 a 的值，在这个值下，后面的表达式最大化 (带任意断开的连接)。贪婪的政策的行动看起来最好在短任期时的 lookahead-according v 一步。通过构造，贪婪策略满足政策改进定理 (4.7) 的条件，因此我们知道它与原始策略一样好，或者比原始策略更好。在原有政策的基础上制定新政策，使其对原有政策的价值功能产生贪欲，这一过程称为政策改进。假设新的贪婪的政策, 吗? 一样好, 但并不比, 老政策 。然后 v = v ?, 从 (4.9), 对所有 年代: v ?(s)= max 一个 E(Rt + 1 + v ?(圣 + 1)| 圣 =,=)
= 最大一个
r s ?, p(s ?r | s,)
r + v ?(?)
. 电

但这是一样的贝尔曼最优性方程 (4.1), 因此,v 吗? 必须 v∗, ? 必须优化政策。因此，除了最初的政策已经是最优的时候之外，政策改进必须给我们一个严格更好的政策。到目前为止，在这一节中，我们已经讨论了确定性策略的特殊情况。在一般情况下, 一个随机策略 指定概率, (|), 在每一个行动, 一个, 在每个州,s。我们不会经历的细节, 但事实上所有的思想这部分很容易扩展到随机的政策。特别地，政策改进定理在随机情况下进行。此外，如果在政策改进步骤 (4.9) 中有联系——也就是说，如果有几个动作达到了最大值——那么在随机情况下，我们不需要从中选择一个动作。相反，每个最大化操作都可以得到在新的贪婪策略中被选择的概率的一部分。任何分配方案都是允许的，只要所有的次最大值动作都是零概率的。图 4.1 的最后一行显示了随机策略的策略改进示例。 , 这里原来的政策是等概率的随机的政策, 新政策, 吗? 关于 v 贪婪。价值函数 v 左下图所示的设置可能的 ? 在右下角的图表中显示。美国有多个箭头在 吗? 图是指几个动作在 (4.9) 中达到最大值的动作; 这些行为之间的概率分配是允许的。任何此类政策的价值函数,v ?(s), 可以被检验是 −1−2, 或者 −3 州, 年代, 而 v (s) 是最多 −14。因此,v ?(s) v (年代)

年代, 说明政策改进。尽管在这种情况下, 新政策 吗? 恰好是最优的，一般来说只有改进是有保证的。

4.3 政策迭代

, 一旦一个策略, 改进了使用 v 产生更好的政策, 吗? 我们可以计算 v 呢? 和改善再取得一个更好的 ? ?。因此，我们可以得到一系列单调改进的政策和价值函数:

0 子邮件 −→v 0 我 −→ 1 子邮件 −→v 1 我 −→ 2 子邮件 −→ • • • 我 −→ ∗ 子邮件 −→v∗

在 E−→ 代表一个政策评估和我 −→ 代表一个政策改进。每个策略都保证比前一个策略有严格的改进 (除非它已经是最优的)。由于有限的 MDP 只有有限数量的策略，因此这个过程必须在有限的迭代中收敛到最优策略和最优值函数。这种寻找最优策略的方法称为策略迭代。完整的算法在下面的框中给出。注意，每个策略评估 (本身就是一个迭代计算) 都是从前一个策略的值函数开始的。这通常会导致政策评估的收敛速度大幅提高 (大概是因为从一个政策到下一个政策的价值函数变化不大)。

例 4.2: 杰克的汽车租赁杰克为一家全国性的汽车租赁公司管理两个地点。每天都有一定数量的顾客到每个地方租车。如果杰克有一辆车的话，他就把车租出去，并被国家公司记入 10 美元。如果他在那个地方没有车，那生意就完了。汽车在归还后的第二天就可以出租了。为了确保车辆可以在需要的地方使用，Jack 可以在两个地点之间过夜，每辆车的费用是 2 美元。我们假设汽车的数量要求在每个位置并返回泊松随机变量, 也就是说, 的概率数字 n = N n !e − , 是预期的数量。假设 3 和 4 租赁请求第一个和第二个位置，返回 3 和 2。略来简化问题, 我们假设可以有不超过 20 辆汽车在每个位置 (全国任何额外的车回到了公司, 因此从问题消失), 最高 5 辆车可以从一个位置移动到另一个晚上。我们把贴现率 = 0.9, 制定持续有限 MDP, 几天的时间步骤, 国家汽车的数量在每个位置在一天结束的时候, 和操作的两个地点之间的净数量的汽车移动过夜。图 4.2 显示了从从不移动任何汽车的策略开始的策略迭代发现的策略序列。

图 4.2: 策略迭代对 Jack 的汽车租赁问题发现的策略序列，以及最终的状态值函数。前五幅图显示，对于一天结束时每个地点的汽车数量，将从第一个位置移到第二个位置的汽车数量 (负数表示从第二个位置转到第一个位置)。每一个连续政策是对以前政策的严格改进，最后一个政策是最优的。

策略迭代常常以令人惊讶的很少的迭代收敛，例如杰克的汽车租赁的例子，并且正如图 4.1 中的例子所示。图 4.1 左下图显示了等概率随机策略的值函数，右下图显示了该值函数的贪心策略。政策改进定理告诉我们，这些政策优于原来的随机政策。然而，在这种情况下，这些策略不仅是更好的，而且是最优的，在最小的步骤中向终端状态前进。在本例中，策略迭代将在一次迭代之后找到最优策略。第 80 页的策略迭代算法有一个微妙的缺陷，即如果策略在两个或两个以上的策略之间持续切换，它可能永远不会终止。这对于教育学来说是可以的，但对于实际应用来说就不行了。修改的伪代码保证收敛。? 练习 4.5 如何为动作值定义策略迭代? 给一个完整的算法计算 q∗, 类似于为计算 v∗80 页。请特别注意这个练习，因为所涉及的思想将贯穿整个练习剩下的书。?

练习 4.6 假设你只局限于考虑 -soft 政策, 这意味着选择每个动作每个状态的概率, 年代, 至少是 / | |。定性地描述在每个步骤 3、2 和 1 中需要的更改，策略迭代算法的秩序,v∗80 页。? 练习 4.7(编程) 为策略迭代编写一个程序，并通过以下更改重新解决 Jack 的汽车租赁问题。杰克的一个员工住在第一个地方，每天晚上乘公共汽车回家，住在第二个地方附近。她很乐意免费把一辆车送到第二个地方。每增加一辆车的价格仍然是 2 美元，所有的车都朝相反的方向行驶。此外，杰克在每个地点都有有限的停车位。如果超过 10 辆车在一个地点过夜 (在任何车辆移动后)，那么使用第二个停车场的额外费用必须为 4 美元 (与保留的车辆数量无关)。这些非线性和任意的动力学经常出现在实际的问题中，不能简单地用动态规划以外的优化方法来处理。要检查您的程序，首先复制给出的结果原始的问题。?

4.4 值迭代

策略迭代的一个缺点是, 它的每个迭代包括政策评估, 这本身就可能是一个旷日持久的迭代计算需要多个横扫状态集。如果政策评估是迭代完成的, 然后完全收敛 v 只发生在极限。我们必须等待精确的汇合，还是我们能不能就此止步? 图 4.1 中的示例显然表明，可以截断政策评估。在这个示例中，超过前三个的策略评估迭代对相应的贪婪策略没有影响。实际上，策略迭代的策略评估步骤可以通过多种方式被截断，而不会失去策略迭代的收敛性保证。一个重

要的特殊

case 是在一次扫描 (每个状态的一次更新) 之后停止策略评估的时候。这种算法称为值迭代。它可以写成一种特别简单的更新操作，结合了策略改进和截断的策略评估步骤:

vk + 1(s)。= 最大一个 E(Rt + 1 + vk(圣 + 1)| 圣 =,=)

= 最大一个

r s ?, p(s ?r | s,)

r + vk(?)

, (4.10)

为所有的 s s 任意 v0, 序列 vk 可以显示在相同的条件下收敛于 v∗ 保证 v∗ 的存在。另一种理解值迭代的方法是参考 Bellman 最优方程 (4.1)。注意，只需将 Bellman 最优方程转换为更新规则，就可以获得值迭代。还要注意，值迭代更新与策略评估更新 (4.5) 是相同的，除非它要求对所有操作都采取最大的操作。查看这种密切关系的另一种方式是在第 59 页 (策略评估) 和图 3.4 左边 (值迭代) 比较这些算法的备份图。这两个是自然计算 v 和 v∗ 备份操作。最后，让我们考虑值迭代是如何终止的。政策评估, 价值迭代正式要求无限的迭代收敛到 v∗。在实践中，当值函数在一次扫描中仅发生少量更改时，我们将停止。下面的框显示了一个具有这种终止条件的完整算法。

价值迭代有效地结合了它的每一个清扫，一扫政策评估和一扫政策改进。更快的收敛性通常是通过在每个策略改进扫描之间插入多个策略评估来实现的。总的来说，完整的截断策略迭代算法可以被看作是扫描序列，其中一些使用策略评估更新，其中一些使用值迭代更新。因为 (4.10) 中的最大操作是两者之间唯一的区别

这些更新，这仅仅意味着最大操作被添加到策略评估的一些扫描中。所有这些算法都收敛到一个最优策略的折现有限 MDPs。

例 4.3: 赌徒的问题，赌徒有机会在抛硬币序列的结果上下注。如果硬币是正面朝上，他赢的钱和他掷硬币所得的钱一样多; 如果是反面，他就会失去赌注。游戏结束时，赌徒以达到 100 美元的目标而获胜，或因钱用光而失败。在每次抛硬币时，赌徒必须以整数的美元来决定自己的资本份额。这个问题可以被表述为一个不打折扣的，偶然的，有限的

99 75 50 25 1 10 20 30 40 50 1 0 0.2 0.4 0.6 0.8 1

资本的资本价值估计

最后的政策 (股份) 扫 1 扫 2 扫 3 扫描 32 终值函数

图 4.3:ph = 0.4 时赌徒问题的解决方案。上面的图显示了 func 的值。通过连续的值迭代扫描发现。的下图显示了最终的策略。MDP。状态是赌徒首都年代 1,2,。99 和操作风险, 0 1。分钟 (100− 年代)。除了那些赌徒达到他的目标 (当目标是 +1 时)，所有转变的奖励都是零。然后状态值函数给出从每个状态中获胜的概率。政策是从资本水平到股权水平的映射。最优政策使达到目标的可能性最大化。让 ph 表示硬币朝上的概率。如果已知 ph 值，那么整个问题就是已知的，可以通过值迭代来解决。图 4.3 显示了在 ph = 0.4 的情况下，值函数在连续的值迭代扫描中发生的变化，以及最终找到的策略。这种政策是最优的，但不是唯一的。实际上，有一个完整的家庭的最优策略，都对应于对最优值函数的 argmax 操作选择的关系。你能猜到整个家庭是什么样子吗?

练习 4.8 为什么赌徒问题的最优策略有如此奇怪的形式? 特别地，对于 50 的资本，它把所有的赌注都押在一次抛掷上，但是对于 51 的资本，它就这样做了不是。为什么这是一个好的政策? ?

练习 4.9(编程) 实现赌徒问题的值迭代，并求解 ph = 0.25 和 ph = 0.55。在编程中，您可能会发现引入两个与终止对应的虚拟状态 (大写 0 和 100)，分别为 0 和 1。图形化地显示结果，如图 4.3 所示。你的结果是稳定 →0 ? ? 练习 4.10 操作值的值迭代更新 (4.10) 的模拟值是什么，qk + 1(,)? ?

4.5 异步动态规划

DP 方法的主要缺点, 到目前为止, 我们已经讨论了它们涉及业务在整个国家的 MDP, 也就是说, 他们需要清洁工的状态。如果国家设置非常大, 甚至一个单一的扫描可以是非常昂贵的。例如，西洋双陆棋有超过 1020 个州。即使我们可以在每秒 100 万个状态上执行值迭代更新，完成一次扫描也需要花费超过 1000 年的时间。异步 DP 算法是现有的迭代 DP 算法，它们不是按照对状态集的系统扫描来组织的。在更新其他状态的值之前，可以更新一些状态的值几次。然而，为了正确地收敛，异步算法必须继续更新所有状态的值: 它不能在计算中某个点之后忽略任何状态。异步 DP 算法允许在选择状态更新时具有很大的灵活性。例如，异步值迭代的一个版本使用值迭代更新 (4.10) 更新每个步骤上只有一个状态 sk 的值 k。如果 0 $<$ 1, 渐近收敛到 v∗ 保证考虑到所有国家只发生在序列 sk 无限次数 (甚至可以随机序列)。(在不打折扣的章节案例中，可能有一些更新顺序不会导致趋同，但相对而言，避免这些更新比较容易。) 类似地，可以混合策略评估和值迭代更新，以生成一种异步截断策略迭代。尽管这个和其他更不常见的 DP 算法的细节超出了这本书的范围，但是很明显，一些不同的更新形成了构建块，可以在各种各样的无甜味 DP 算法中灵活地使用。当然，避免扫描并不一定意味着我们可以减少计算量。这仅仅意味着，算法在改进策略之前，不需要陷入任何无望的长扫描。我们可以尝试利用这种灵活性，选择我们应用更新的状态，以提高算法的进展速度。我们可以尝试对更新进行排序，让价值信息以一种有效的方式从一个状态传播到另一个状态。有些国家可能不需要像其他国家那样经常更新它们的价值观。如果它们与最佳行为无关，我们甚至可以尝试跳过更新一些状态。第 8 章讨论了实现这一点的一些想法。异步算法还使计算与实时交互更容易混合。为了解决给定的 MDP，我们可以在代理实际体验 MDP 的同时运行迭代 DP 算法。代理的经验可以使用

确定 DP 算法应用其更新的状态。同时，DP 算法的最新值和策略信息可以指导代理的决策。例如，我们可以在代理访问状态时对状态应用更新。这使得将 DP 算法的更新集中到与代理最相关的状态集的某些部分成为可能。这种聚焦是强化学习的重复主题。

4.6 广义政策迭代

策略迭代包括两个同步的、相互作用的过程，一个使值函数与当前策略 (策略评估) 一致，另一个使策略对当前值函数 (策略改进) 贪婪。在策略迭代中，这两个过程交替进行，每个过程在另一个过程开始之前完成，但这并不是真正必要的。例如，在值迭代中，每次策略改进之间只执行一次策略评估迭代。在异步 DP 方法中，评估和改进过程以更细的粒度进行交叉。在某些情况下，一个进程在返回到另一个进程之前更新一个状态。只要这两个进程继续更新所有状态，最终的结果通常是对最优值函数和最优策略的相同收敛。

评价

改进 吗? 贪婪 (V) V V ?v

v∗ ∗ 我们使用广义策略迭代 (GPI) 一词来表示让策略评估和策略改进过程相互作用的总体思想，而不依赖于这两个过程的粒度和其他细节。几乎所有的强化学习方法都被很好地描述为 GPI。也就是说，所有的策略和值函数都具有可识别性，策略总是相对于值函数进行改进，而值函数总是被驱动到策略的值函数，如右边的图所示。如果评价过程和改进过程都稳定，即不再产生变化，那么价值函数和政策必须是最优的。只有当它与当前政策一致时，价值函数才会稳定，并且只有当它对当前值函数贪婪时，才会稳定。因此，只有当发现一个策略对其自身的评估函数是贪婪的时，这两个进程才会稳定。这意味着 Bellman 最优性方程 (4.1) 成立，从而使策略和值函数是最优的。GPI 中的评估和改进过程可以看作是竞争和合作的过程。它们竞争的意义在于它们向相反的方向拉。使策略对值函数变得贪婪通常会使值函数对更改的策略不正确，使值函数与策略保持一致通常会导致策略不再贪婪。然而，从长期来看，这两个过程相互作用，以找到一个共同的解决方案: 最优值函数和最优策略。

v∗, ∗ v, 我们也可以将 GPI 中评估和改进过程之间的相互作用考虑为两个约束或目标——例如，如右边的图表所建议的二维空间中的两条线。虽然实际的几何图形要比这个复杂得多，但图表显示了真实情况下会发生什么。每个进程都将值函数或策略驱动到一个表示解决方案的行中。两个目标中的一个。目标相互作用是因为这两条线不是正交的。直接向一个目标前进会导致一些运动偏离另一个目标。但不可避免的是，联合过程更接近最优性的总体目标。这个图中的箭头与策略迭代的行为相对应，每个箭头都带着系统完成两个目标中的一个。在 GPI 中，你也可以对每个目标采取更小、不完全的步骤。在任何一种情况下，这两个过程共同实现了最优性的总体目标，尽管两者都没有试图直接实现这一目标。

4.7 动态编程效率

DP 可能并不适用于非常大的问题，但是与其他解决 MDPs 的方法相比，DP 方法实际上是非常有效的。如果我们忽略了一些技术细节，那么 (最坏的情况)DP 方法找到最优策略的时间是状态数和动作数的多项式。如果 n 和 k 表示状态和动作的数量, 这意味着 DP 方法需要大量的计算操作, 小于 n 和 k 的多项式函数。DP 方法保证在多项式时间内找到最优政策尽管 (确定性) 政策是 kn 的总数。从这个意义上说，DP 比任何直接的政策空间搜索都要快得多，因为直接搜索必须详尽地检查每一个政策才能提供同样的保证。线性规划方法也可用于解 MDPs，在某些情况下，它们的最坏情况收敛保证比 DP 方法更好。但是线性规划方法在比 DP 方法更少的状态下变得不切实际 (大约是 100 倍)。对于最大的问题，只有 DP 方法是可行的。由于维度的诅咒，有时人们认为 DP 的适用性是有限的，因为状态的数量常常随状态变量的数量呈指数增长。大型的状态集确实会造成困难，但是这些都是问题的固有困难，而不是开发计划署作为解决方案的方法。事实上，相对于直接搜索和线性规划等竞争方式，DP 更适合处理大型状态空间。在实践中，DP 方法可以与当今的计算机一起使用，来解决具有数百万个状态的 MDPs。策略迭代和值迭代都得到了广泛的应用，而且不清楚在一般情况下，哪种方法更好。在实践中，这些方法的收敛速度通常比理论的最坏情况运行时间快得多，尤其是当它们启动时

具有良好的初始值函数或策略。对于状态空间较大的问题，通常首选异步 DP 方法。要完成同步方法的一次扫描，需要对每个状态进行计算和内存。对于某些问题，即使如此多的内存和计算是不切实际的，但是这个问题仍然是可以解决的，因为沿着最优解轨迹出现的状态相对较少。在这种情况下可以应用异步方法和 GPI 的其他变体，并且可以比同步方法更快地找到好的或最优的策略。

4.8 总结

在本章中，我们已经熟悉了动态规划的基本思想和算法，因为它们与求解有限的 MDPs 有关。策略评估是指 (典型地) 对给定策略的值函数进行迭代计算。策略改进是指根据策略的价值函数计算改进的策略。将这两个计算放在一起，我们得到了策略迭代和值迭代，这是两种最流行的 DP 方法。其中任何一种都可以用于可靠地计算有限 MDPs 的最优策略和值函数，并提供 MDP 的完整知识。经典的 DP 方法在状态集中执行扫描操作，对每个状态执行预期的更新操作。每个这样的操作根据所有可能的继承状态的值及其发生的可能性更新一个状态的值。预期的更新与 Bellman 方程密切相关: 它们只不过是将这些方程转化为赋值语句。当更新不再导致值的任何变化时，满足相应 Bellman 方程的值出现收敛。就像有四个主要价值函数 (v ,v∗、q 和 q∗), 有四个相应的贝尔曼方程和四个预期相应更新。DP 更新操作的直观视图由它们的备份图提供。通过将 DP 方法视为通用策略迭代 (GPI)，可以深入了解几乎所有的增强学习方法。GPI 是围绕一个近似策略和一个近似值函数的两个相互作用过程的一般思想。一个过程接受给定的策略并执行某种形式的策略评估，将值函数更改为更类似于策略的真正值函数。另一个过程接受给定的值函数并执行某种形式的策略改进，更改策略以使其更

好，假设值函数是其值函数。虽然每个过程都改变了另一个过程的基础，但总的来说，它们共同努力找到一个共同的解决方案: 策略和价值函数，这两个过程都没有改变，因此是最优的。在某些情况下，GPI 可以被证明为收敛的，尤其是我们在本章中介绍的传统 DP 方法。在其他情况下，收敛性还没有得到证明，但是 GPI 的思想仍然提高了我们对这些方法的理解。不需要完全遍历状态集来执行 DP 方法。异步 DP 方法是一种就地的迭代方法，以任意顺序更新状态，可能是随机确定的，并使用过时的信息。

5. 第五章蒙特卡罗方法

在本章中，我们考虑了我们的第一个评估价值函数和发现最优策略的学习方法。与前一章不同的是，这里我们不假设对环境有完整的了解。蒙特卡罗方法只需要与环境进行实际或模拟交互时的状态、动作和奖励的经验样本序列。从实际经验中学习是惊人的，因为它不需要事先了解环境的动态，但仍然可以获得最佳行为。从模拟经验中学习也很有用。虽然需要一个模型，但是模型只需要生成示例转换，而不需要生成动态编程所需的所有可能转换的完整概率分布。令人惊讶的是，在许多情况下，根据期望的概率分布很容易生成经验，但以显式的形式获取分布是不可行的。蒙特卡罗方法是一种基于平均样本收益率的增强学习方法。为了确保定义良好的回报是可用的，这里我们定义蒙特卡罗方法仅用于情景任务。也就是说，我们假设经验被分为不同的阶段，所有的阶段最终都结束了，不管选择什么动作。只有在事件完成时，价值评估和策略才会改变。因此，蒙特卡罗方法可以循序渐进地逐步进行，但不是一步一步 *(*在线*)* 的意义。“*Monte Carlo*”一词通常用于任何估计方法，其操作涉及一个重要的随机组件。在这里，我们特别将它用于基于平均完整回报的方法 *(*与从部分回报中学习的方法相反，我们将在下一章中进行讨论*)*。蒙特卡罗方法样本和每个状态-动作对的平均回报就像我们在第二章的样本和每个动作的平均回报一样。主要的区别是，现在有多个状态，每个状态都像一个不同的强盗问题 *(*比如关联搜索或上下文强盗*)*，而不同的强盗问题是相互关联的。也就是说，在一个州采取行动后的回报取决于在同一事件中在后来的州采取的行动。由于所有的动作选择都在进行学习，从早期状态的角度来看，问题变得非平稳的。

为了处理非平稳性，我们采用了第 *4* 章为 *DP* 开发的通用策略迭代 *(GPI)* 的思想。这里我们从 *MDP* 知识中计算值函数，这里我们从 *MDP* 的样本返回中学习值函数。值函数和相应的策略仍然以基本相同的方式相互作用以获得最优性 *(GPI)*。*DP* 的章, 我们首先考虑预测的问题 *(v* 和 *q* 固定的计算任意政策 *)* 然后政策改进„ 最后, 通过谷歌控制问题及其解决方案。从 *DP* 中获得的每一个想法都扩展到蒙特卡罗的情况，在这种情况下只有样本经验可用。

5.0.1 蒙特卡罗预测

我们首先考虑蒙特卡罗方法来学习给定策略的状态值函数。回想一下，一个状态的值是预期的回报预期累积未来折扣奖励——从这个状态开始。从经验中估计它的一个明显的方法，就是简单地对访问该国后观察到的收益进行平均。当观察到更多的回报时，平均值应该收敛到期望值。这个想法是所有蒙特卡罗方法的基础。特别是, 假设我们希望估计 v (s) 的价值状态下政策 , 给定一组集通过 和通过年代。每个发生的状态在一集被称为访问年代。当然, 年代可能多次访问相同的事件; 让我们叫它第一次访问在一个事件的首次访问。第一次访问 MC 方法估计 v (s) 的平均回报后第一个访问, 而平均每次访问 MC 方法后返回所有访问。这两个蒙特卡罗 (MC) 方法非常相似, 但是略有不同的理论属性。第一次拜访 MC 的研究最为广泛，可以追溯到 20 世纪 40 年代，这也是我们在本章关注的重点。每一次访问 MC 都更自然地扩

展到函数逼近和资格跟踪，如第 9 章和第 12 章所讨论的那样。第一次参观的 MC 以程序形式显示在方框中。每一次拜访的主持人都是一样的，除了没有检查 St 是否在之前发生过。

第一次访问 MC 和每次访问 MC 收敛于 v (s) 访问的数量 (或第一次访问) 年代趋于无穷。这是第一次访问的情况下容易看到 MC。在这种情况下每个返回是一个独立同分布的估计 v (s) 与有限方差。根据大数定律，这些估计的平均数序列收敛于它们的期望值。每个平均本身就是一个无偏估计, 其误差的标准差为 1 /√n, 其中 n 是返回平均的数量。每次访问 MC 并不简单, 但其估计也成平方收敛 v (s)(辛格和萨顿,1996)。通过一个例子可以很好地说明蒙特卡罗方法的使用。例 5.1:Blackjack 流行的赌场纸牌游戏的目标是在不超过 21 的情况下，获得数值尽可能大的纸牌的总和。所有的牌都是 10 张，ace 可以是 1 张或 11 张。我们考虑的是每个玩家与经销商独立竞争的版本。游戏开始时，发牌者和牌手各发两张牌。其中一张牌是正面朝上的，另一张是正面朝下的。如果玩家马上有 21 张牌 (一张王牌和一张 10 张牌)，这就叫做自然牌。然后他就赢了，除非发牌人也有自然牌，在这种情况下，游戏是平局。如果玩家没有一个自然牌，他可以要求额外的牌，一个接一个 (点击)，直到他停止 (粘贴) 或超过 21(崩溃)。如果他破产了，他就输了; 如果他坚持下去，就该轮到交易商了。经销商按照固定的策略击球或坚持，没有选择的余地: 他坚持任何 17 或以上的点数，并击中其他。如果经销商破产，那么玩家获胜; 否则，结果-赢、输或取-取决于谁的最终总数接近 21。玩 blackjack 自然是一种情景式的有限 MDP。21 点的每一场比赛都是一集。奖励 + 1−1 和 0 的胜利, 失去, 分别和绘画。游戏内的所有奖励都是零, 我们不要折扣 (= 1); 因此，这些终极奖励也是回报。玩家的动作是击中或卡住。状态取决于玩家的牌和经销商的显示卡。我们假设牌是从无限的牌牌 (即:)，以便对已处理过的卡片进行跟踪没有任何好处。如果玩家手里拿着一张他可以数到 11 的王牌而不被击出，那么这张王牌就是可用的。在这种情况下，它总是被算作 11，因为把它算作 1 就等于 11 或更少，在这种情况下，没有必要做出决定，因为显然，玩家应该总是命中。因此，玩家在三个变量的基础上做出决定: 当前的总数 (12-21)，经销商的一张展示卡 (ac - 10)，以及他是否拥有一个可用的 ace。总共有 200 个州。考虑如果玩家的总数是 20 或 21，如果不这样做的话，该策略就会失效。为了通过蒙特卡罗方法找到该策略的状态值函数，一个模拟了许多使用策略的 blackjack 游戏，并平均每个状态下的返回值。通过这种方式，我们得到了图 5.1 所示的状态值函数的估计值。使用 ace 的状态的估计不那么确定，也不那么规则，因为这些状态不太常见。无论如何，在 50 万场比赛后，价值函数是非常接近的。

10000 集之后

500000 集之后

图 5.1: 由蒙特卡罗策略评估计算的仅停留在 20 或 21 上的 21 点的 21 点策略的近似状态值函数。

练习 5.1 考虑图 5.1 中右边的图表。为什么估计值函数会在后面的两行中跳起来? 为什么它在左边的最后一行会下降? 为什么上面的图中最前面的值更高? 比低吗? ?

练习 5.2 假设在 21 点任务中使用了每个访问 MC 而不是第一次访问 MC。你认为结果会有很大不同吗? 为什么或为什么不呢??

虽然我们对 blackjack 任务中的环境有完整的了解，但是使用 DP 方法来计算值函数并不容易。DP 方法需要下一个事件的分布—特别是，它们需要四参数函数 p 所给出的环境动力学—而且对于 21 点来说，要确定这一点并不容易。例如，假设玩家的和是 14，他选择坚持。他终止的概率是 +1 作为经销商的显示卡的函数? 在应用 DP 之前，必须计算所有的概率，而这种计算通常是复杂的，容易出错的。相反，生成蒙特卡罗方法所需的示例游戏是很容易的。这种情况经常发生; 蒙特卡罗方法能够单独使用样本集，这是一个非常重要的优势，即使我们已经完全了解了环境的动态。我们能把备份图的概念推广到蒙特卡罗算法吗? 备份图的一般思想是在顶部显示要更新的根节点，并在下面显示所有的转换和叶节点，它们的奖励和估计值对更新有贡献。蒙特卡罗估计 v , 根节点是一个状态, 下面是过渡的整个轨迹沿着一个特定的一集, 结局

在终端状态，如右边所示。DP 图 (第 59 页) 显示了所有可能的转换，而蒙特卡罗图只显示了在一集中抽样的转换。而 DP 图只包含一步转换，蒙特卡罗图一直到这一集的结尾。图

中的这些差异准确地反映了算法之间的基本差异。蒙特卡罗方法的一个重要事实是每个状态的估计是独立的。一个国家的估计数不以任何其他国家的估计数为基础，如 DP 的估计数。换句话说，蒙特卡罗方法不像我们在前一章中定义的那样引导。特别要注意，估计单个状态值的计算开销与状态数无关。这可以使蒙特卡罗方法在只需要一个或一个子集的值时特别具有吸引力。的状态。可以从感兴趣的状态开始生成许多样例片段，只对这些状态的收益进行平均，忽略其他所有状态。这是蒙特卡罗方法的第三个优势，它可以使用 DP 方法 (在从实际经验和模拟经验中学习的能力之后)。

线圈上的气泡。Hersh and Griego(1969)。与许可转载。c ?1969《科学美国人》,《自然美国公司》，版权所有。例 5.2: 肥皂泡假设一个线框形成一个封闭的环，在肥皂水里泡，形成一个肥皂表面或气泡符合它的边缘到线框。如果线框的几何形状是不规则但已知的，那么如何计算曲面的形状呢? 形状具有这样的性质: 相邻点施加在每个点上的总力为 0(否则形状会改变)。这意味着任何一点上的表面高度都是在这个点周围的小圆上的高度的平均值。此外，表面必须满足其边界与线框。处理此类问题的通常方法是在所覆盖的区域上设置一个网格在网格点上通过迭代计算得到其高度。边界上的网格点被强制到线框，其他所有的点都被调整到它们四个最近邻的平均高度。然后这个过程迭代，就像 DP 的迭代策略评估一样，最终收敛到期望的表面。这类似于蒙特卡罗方法最初设计的问题。与上面描述的迭代计算不同，想象一下站在地面上，随机行走，从网格点随机走向相邻的网格点，概率相等，直到你到达边界。结果表明，边界处高度的期望值是接近起始点理想表面高度的近似值 (实际上，它正是上面描述的迭代法计算的值)。因此，我们可以近似地估计出

通过简单地平均许多次行走的边界高度，在一点开始。如果只关心某一点上的值，或任何固定的小点集，那么蒙特卡罗方法可以比基于局部一致性的迭代方法有效得多。

5.1 行为值的蒙特卡罗估计

如果一个模型不可用，那么估计动作值 (状态-动作对的值) 而不是状态值是特别有用的。对于模型，状态值本身就足以确定策略; 我们只需要向前看一步，然后选择任何能带来奖励和下一个状态的最佳组合的行为，就像我们在关于 DP 的章节中所做的那样。但是，如果没有模型，仅使用状态值是不够的。必须明确地估计每个操作的值，以便这些值在建议策略时有用。因此, 我们的一个主要目标估计问 ∗ 蒙特卡罗方法。为此，我们首先考虑行动价值的政策评估问题。行动值的政策评估问题是估计 q (,), 开始时的预期收益在国家年代, 采取行动, 然后遵循政策 。蒙特卡罗方法在本质上和描述国家价值是一样的，除了现在我们讨论的是对国家行动的访问而不是对国家的访问。一个状态-动作对，据说在一集中被访问，如果有任何一个状态被访问并且在其中被采取行动。每一次访问 MC 方法估计状态-动作对的值，作为所有访问后返回的平均值。第一次访问 MC 方法在每一集访问状态并选择动作的第一次之后平均返回。这些方法与以前一样，以四倍的方式收敛于实际期望值，因为每个状态-动作对的访问次数趋于无穷大。唯一的复杂之处在于许多状态-动作对可能永远不会被访问。如果 是一个确定性的政策, 然后在 人将只返回后从每个国家的行动。在没有平均回报的情况下，蒙特卡罗对其他行动的估计不会因经验而有所改善。这是一个严重的问题，因为学习动作值的目的是帮助在每个状态中选择可用的动作。为了比较替代方案，我们需要估计来自每个状态的所有操作的值，而不仅仅是我们当前喜欢的。这是在第 2 章 k -武装土匪问题上下文中讨论的维持勘探的一般问题。为了使政策评估工作的行动价值，我们必须保证持续的探索。一种方法是指定以状态-动作对开始，并且每对都有非零的被选择为开始的可能性。这保证了所有的状态-动作对将被访问无限次，在无限次的集数中。我们称之为探索开始的假设。探索开始的假设有时是有用的，但通常不能依赖它，特别是当从与环境的实际交互中直接学习时。在这种情况下，启动条件不太可能有这么大的帮助。确保遇到所有状态操作对的最常见的替代方法是

只考虑具有非零概率的随机策略，在每个状态中选择所有操作。我们将在后面的小节中讨论这种方法的两个重要变体。目前，我们保留了探索开始和完成完整的蒙特卡罗控制方法的

假设。练习 5.3 备份是什么图蒙特卡罗估计 q 吗??

5.2 蒙特卡罗控制

我们现在准备考虑如何在控制中使用蒙特卡罗估计，即近似最优策略。总体思路是按照与 DP 章节相同的模式进行，也就是说，按照广义政策迭代的思想进行评价

改进 问 吗? 贪婪 (Q) 问?q (GPI)。在 GPI 中，一个函数同时维护一个近似策略和一个近似值函数。值函数被反复修改以更接近当前策略的值函数，而策略在当前值函数方面被反复改进，如图右所示。这两种变化在某种程度上是相互作用的，因为每个变化都为另一个创建了一个移动的目标，但它们共同导致策略和价值函数趋于最优。首先，让我们考虑一下经典策略迭代的蒙特卡罗版本。在这种方法中, 我们表现的交替政策评估和政策改进的完整步骤, 从任意政策 0 开始和结束的最优策略和最优行为价值功能: 0 子邮件 −→q 0 我 −→ 1 子邮件 −→q 1 我 −→ 2 子邮件 −→ ••• 我 −→ ∗ 子邮件 −→ 问 ∗

在 E−→ 表示一个完整的政策评估和我 −→ 表示一个完整的政策改进。政策评估完全按照前一节所述进行。由于近似的动作-值函数渐近地逼近真函数，因而出现了许多次发作。现在，让我们假设我们确实观察了无数的情节，而且，这些情节是在探索开始时产生的。根据这些假设, 蒙特卡洛方法将计算每个 q k, 任意 k。策略改进是通过使策略对当前值函数变得贪婪来实现的。在这种情况下，我们有一个动作值函数，因此不需要模型来构造贪婪策略。对于任何行为价值函数 q, 相应的贪婪的政策是, 对于每个 年代, 确定性与最大操作值选择一个动作:

(年代)。= argmax 一个问 (年代)。(5.1)

政策改进然后可以通过构建每个 k + 1 对 q k 贪婪的政策。政策改进定理 (4.2 节), 那么适用于 k

和 k + 1, 因为对于所有 年代, q k(年代, k + 1(s))= q k(年代,argmax 一个 q k(s))

= 最大一个 q k(年代) q k(年代, k(s)) v k(年代)。正如我们在前一章中讨论的, 这个定理保证我们每个 k + 1 均匀比 k, 或 k 一样好, 在这种情况下, 他们都是最优策略。这反过来又保证了整个过程收敛到最优策略和最优值函数。在这种情况下，蒙特卡罗方法可以用来寻找最优的策略，只提供样本集，而不了解环境的动力学知识。为了方便地得到蒙特卡罗方法的收敛保证，我们做了两个不太可能的假设。一个是，这些事件有探索的开始，另一个是，政策评估可以用无数个事件来完成。为了得到一个实用的算法，我们必须去除这两个假设。我们将第一个假设的考虑推迟到本章后面的部分。目前，我们关注的是，政策评估的作用是无限次的。这个假设相对容易删除。事实上，即使在经典的 DP 方法中也会出现同样的问题，比如迭代策略评估，它也只渐进地收敛于真值函数。在 DP 和蒙特卡罗的情况下，有两种方法可以解决这个问题。一个是持有公司的想法近似 q k 在每个政策评估。测量和假设是为了获得估计误差的大小和概率的界限，然后在每次政策评估期间采取充分的步骤，以确保这些界限足够小。这种方法可能完全令人满意，因为它可以保证正确的收敛到某种程度的近似。然而，它也可能需要太多的插曲，在实践中有用的任何问题，除了最小的问题。第二种方法是避免在名义上需要进行政策评估的无数个阶段，在这些阶段中，我们放弃了在恢复政策改进之前完成政策评估的努力。在每个评估步骤, 我们将向 q k 价值函数, 但实际上我们并不期望接近除了在许多步骤。我们在第 4.6 节首次介绍 GPI 的概念时使用了这个概念。该思想的一种极端形式是价值迭代，即在策略改进的每个步骤之间只执行一次迭代策略评估。值迭代的就地版本甚至更为极端; 在那里，我们在单个状态的改进和评估步骤之间交替进行。对于蒙特卡罗的政策评估来说，很自然地要在评估和改进之间交替进行。在每一集结束后，观察到的收益被用于政策评估，然后在每一集访问的所有州都对政策进行改进。一个完整的简单算法沿着这些线，我们称为蒙特卡罗 ES，为蒙特卡罗探索开始，给出了伪代码在盒子的下一页。

蒙特卡洛 ES 的伪代码效率很低，因为对于每个状态-动作对，它维护一个所有返回的列表并反复计算它们的平均值。使用类似于 2.4 节中解释的技术来维护仅仅是平均值和计数 (对于

每个状态操作对)，并且增量地更新它们将会更有效。描述如何修改伪代码以实现这一点。? 在蒙特卡洛 ES 中，每个状态-行为对的所有回报都是累积和平均的，不管观察到什么政策是有效的。很容易看出蒙特卡罗不可能收敛于任何次优策略。如果它这样做了，那么值函数将最终收敛到该策略的值函数，这反过来将导致策略的更改。只有当政策和价值函数都是最优时，才能实现稳定性。当动作-值函数的变化随着时间的推移而减小时，收敛到这个最优不动点似乎是不可避免的，但是还没有得到正式的证明。在我们看来，这是强化学习中最基本的开放理论问题之一 (参见 tsiklis, 2002)。例子 5.3: 解决 21 点的问题将蒙特卡洛 ES 应用到 21 点很简单。因为每一集都是模拟游戏，所以很容易安排包括所有可能性的探索开始。在这种情况下，你只需选择发牌人的牌，玩家的总数，以及玩家是否有可用的 ace，所有这些都是随机的，概率相等。作为初始策略，我们使用前一个 21 点示例中评估的策略，该策略只适用于 20 或 21。对于所有状态-动作对，初始动作-值函数可以为零。图 5.2 显示了蒙特卡洛发现的 21 点的最优策略。该策略与 Thorp(1966) 的“基本”策略相同，唯一的例外是可用 ace 策略中最左的一个等级，而 Thorp 的策略中没有这一项。我们不确定产生这种差异的原因，但是我们相信这里显示的确实是我们所描述的 21 点版本的最佳策略。

One hundred.

V *

1 1

1 1

图 5.2 蒙特卡洛 ES 发现的 21 点的最优策略和状态值函数。所示的状态值函数是从蒙特卡罗 ES 发现的动作值函数中计算出来的。

5.2.1 无探测启动的蒙特卡罗控制

我们怎样才能避免不太可能的探险开始的假设呢? 确保无限次地选择所有操作的惟一通用方法是代理继续选择它们。有两种方法可以确保这一点，这导致了我们所称的“政策上的方法”和“政策外的方法”。策略方法试图评估或改进用于决策的策略，而策略方法则评估或改进与用于生成数据的策略不同的策略。上面开发的蒙特卡罗 ES 方法是一个政策方法的例子。在本节中，我们将展示如何设计一种不使用不切实际的探索开始假设的策略蒙特卡罗控制方法。下一节将考虑策略外方法。在政策控制方法的政策通常是柔软, 这意味着 (|)> 0 年代和所有 (s), 但渐渐地转移越来越接近确定性的最优政策。第 2 章中讨论的许多方法提供了实现这一点的机制。我们目前的政策方法这一节使用 -greedy 政策, 这意味着大多数时候他们选择一个行动, 最大估计行动的价值, 但概率 他们而不是随机选择一个行动。即所有 nongreedy 行动给出的最小概率选择, (s)| |, 剩下的大部分的概率,1− + (s)| |, 贪婪的行动。 -soft 政策的例子, -greedy 策略定义为政策的 (|) | 一个 (s)| 对于所有国家和动作, 对于一些 > 0。在 -soft 政策, -greedy 政策在某种意义上

这是最接近贪婪的。在政策上蒙特卡罗控制的总体思路仍然是 GPI。就像在蒙特卡罗 ES 中一样，我们使用第一次访问 MC 方法来估计当前策略的行动价值函数。但是，如果没有开始探索的假设，我们就不能简单地通过对当前值函数的贪婪来改进策略，因为这样可以防止对非贪婪行为的进一步探索。幸运的是，GPI 并不要求将该策略一直应用到贪婪的策略上，而只是要求将其转向贪婪的策略。在我们的政策方法我们将只有一个 -greedy 政策。对于任何 -soft 政策, , 任何关于 q -greedy 政策是保证优于或等于 。完整的算法在下面的框中给出。

任何 -greedy 政策对 q 是一个比任何 -soft 政策是保证政策的改进公式。让 吗? -greedy 政策。政策改进定理的条件, 因为申请任何 年代:

q (年代, ?(s))=

一个 ?(|)q (,)

= | |(s)

一个 q (s)+(1−) 马克斯一个 q (年代) (5.2)

(s)| |

一个 q (s)+(1−)
一个 (|)− (s)| | 1− q (年代)
(和是一个非负权和为 1 的加权平均数，因此
必须小于或等于最大的平均数)
= | |(s)
一个 q (s)− (s)| |
一个 q (s)+
一个 (|)q (,)
= v (s)。因此, 通过政策改善定理, 吗? (即。,v ?(s) v (s), 对所有 年代)。我们现在证明平等可以容纳只有当两个 吗? 和 -soft 之间的最优政策, 也就是说, 当它们优于或等于其它所有 -soft 政策。考虑一个新的环境, 就像原始的环境中, 除了要求政策是 -soft“内部”环境。新环境具有与原始环境相同的动作和状态集，其行为如下。如果在状态和采取行动, 那么概率 1− 新环境的行为就像旧的环境。随机概率 重新选择英雄的行动, 以同样的概率, 然后像旧的与新的环境, 随机行动。最好可以做在这个新的环境与一般的政策是一样的最好的一个可以在原始环境 -soft 政策。让 v∗

问 ∗ 表示最优值函数的新环境。v∗。v∗ 我们知道这是唯一的解决方案
v∗(s)=(1−) 马克斯一个
问 ∗(s)+ (s)| | 一个
问 ∗(年代) =(1−) 马克斯一个
r s ?, p(s ?
r + v∗(s ?) +

| |(s) 一个 r s ?,
p(s ? r | s,)
r +
当拥有平等和 -soft 政策不再是改进, 然后我们也知道, 从 (5.2) v (s)=(1−) 马克斯
一个 q (s)+ (s)| |
一个 q (年代)
=(1−) 马克斯一个 r s ?,
p(s ?
r + v (?) +
| |(s)
一个 r s ?,p(s ?
r + v (?)

然而, 这个方程是与前一个相同, 除了 v 的替换 v∗。因为 v∗ 是独特的解决方案, 它必须 v =v∗。

大致与前一节相同。现在我们只有 -soft 政策实现最好的政策, 但另一方面, 我们已经消除了假设的探索的开始。

5.2.2 通过重要性抽样进行政策外预测

所有的学习控制方法都面临着一个两难的境地: 它们寻求在后续最优行为条件下学习行为值，但为了探索所有的行为 (寻找最优行为)，它们需要非最优行为。他们如何在执行探索性策略时了解最优策略? 上一节中的 on-policy 方法实际上是一种折衷方法——它学习的行为值不是针对最优策略，而是仍然在探索的接近最优策略。一个更直接的方法是使用两个策略，一个是学习到的，一个是最优策略，一个是探索性的，用于生成行为。正在学习的策略称为目标策略，用于生成行为的策略称为行为策略。在这种情况下，我们说学习是从目标策略的“off”数据中学习，整个过程被称为“off -policy learning”。在本书的其余部分中，我们同时考虑政

策上的方法和政策外的方法。On-policy 方法通常更简单，首先考虑。Off-policy 方法需要额外的概念和符号，而且由于数据是由不同的策略导致的，所以 Off-policy 方法的差异通常更大，收敛速度也更慢。另一方面，政策之外的方法更为强大和普遍。它们包括策略方法，作为目标和行为策略相同的特殊情况。策略外方法在应用程序中也有各种附加用途。例如，它们通常可以应用于从传统的非学习控制器生成的数据或从人类专家那里学习。政策之外的学习也被一些人视为学习世界动态多步骤预测模型的关键 (见第 17.2 节; 萨顿,2009; 萨顿 et al .,2011)。在本节中，我们将从预测问题开始研究非策略方法，其中目标策略和行为策略都是固定的。即, 假设我们希望估计 v 或 q , 但我们是集后另一项政策 b,b 在哪里? = 。在这种情况下, 是目标政策,b 是行为的政策, 这两项政策被认为是固定的,。为了使用集从 b 为 估算值, 我们要求每一个行动在 也是, 至少偶尔, 在 b。也就是说, 我们要求 (|)> 0 意味着 b(|)> 0。这叫做承保范围假设。报道可以看出,b 必须随机州 是不相同的。 目标政策, 另一方面, 可能是确定的, 事实上, 这是一个特别感兴趣的控制应用程序。在控制中，目标策略通常是与当前操作值函数估计值相关的确定性贪婪策略。这一政策成为一个确定性的最优政策政策仍然是随机和更多的探索性行为, 例如, 一个 -greedy 政策。然而, 在这一节中, 我们考虑的预测问题, 在 是不变的。几乎所有的非策略方法都使用重要性抽样，这是一种通用的方法

估计一个分布下的期望值，给出另一个分布下的样本值。我们将重要性抽样应用到非政策学习中，根据目标和行为政策下所发生的轨迹的相对概率来加权收益，称之为进口-抽样比率。给定一个初始状态 St，下一个状态运动轨迹的概率，在 St+1 处，在 +1 处…圣, 发生在任何政策

PrAt, St+1, At+1，…, 圣 | 圣:T−1 在 | = (St)p(St + 1 | 圣, 在 (+ 1 | 圣 + 1) • • • p(St | 圣 −1−1)= T−1 ?k = t (Ak | Sk)p(Sk + 1 | Sk,Ak),

其中 p 为 (3.4) 定义的状态跃迁概率函数。因此，目标和行为策略下轨迹的相对概率 (重要性-采样比) 为

t:T−1 。= ? T−1 k = t (Ak | Sk)p(Sk + 1 | Sk,Ak)? t−1 k = t b(Ak | Sk)p(Sk + 1 | Sk,Ak) = T−1 ?k = t (Ak | Sk) b(Ak | Sk)。(5.3) 虽然轨迹概率依赖于一般未知的 MDP 的跃迁概率，但它们在分子和分母上都是相同的，因此抵消了。重要性抽样比率最终只取决于两个策略和序列，而不是 MDP。回想一下，我们希望估计目标策略下的预期回报 (值)，但由于行为策略，我们所有的都是返回 Gt。这些返回错误的期望 E[Gt | 圣 = s]= vb(s) 所以不能平均获得 v 。这就是重要性抽样的原因。比 t:T−1 转换返回正确的期望值:

E[t:T−1 gt | 圣 = s]= v (年代)。(5.4) 现在我们已经准备好给蒙特卡罗算法的平均回报一批观察发作后的政策 b 估计 v (年代)。在这里，以一种跨事件边界增加的方式对时间步进行编号是很方便的。也就是说，如果第一个片段在时间 100 结束时处于终端状态，那么下一个片段在时间 t = 101 开始。这使我们能够使用时间步数来引用特定事件中的特定步骤。特别是，我们可以定义访问状态 s 的所有时间步骤的集合，表示为 T(s)。这是为每一次访问的方法; 对于第一次访问的方法，T(s) 将只包含第一次访问 s 的时间步骤。同时, 让 T(T) 表示的终止时间 T 后, 第一次和 Gt 表示返回后 T 通过 T(T) 然后 Gt T (s) 是属于国家的回报, 和 t:T(T)−1 ？ T (s) 是相应的重要性抽样比率。估计 v (s), 我们只是规模收益的比率和平均结果:

当重要抽样以这种方式进行时，它被称为普通重要性抽样。一个重要的替代方法是加权重要性抽样，它使用加权平均数，定义为

或者分母为零。理解这两个品种的重要性抽样, 考虑他们的第一次访问方法的估计后观察一个返回的状态。在加权平均估计, 比 t:T(T)−1 单返回取消的分子和分母, 所以估计等于观察恢复独立的比率比非零 (假设)。考虑到这回来是唯一一个注意到, 这是一个合理的估计, 但其预期是 vb(s), 而不是 v (s), 并在统计意义上是有偏见的。相比之下, 第一次访问普通版本的重要性抽样估计量 (5.5) 总是 v (s) 的期望 (公正), 但它可以极端。假设比率为 10，说明在目标策略下观察到的轨迹是行为策略下的 10 倍。在这种情况下，通常的重要性抽样估计是观测到的收益的 10 倍。也就是说，即使这段插曲的轨迹被认为是非常具有代表性的目标政策，它也远未达到预期的回报。在形式上，两种重要性抽样的第一次抽样方法的差异表现在它们的

偏差和方差上。普通的重要性抽样是无偏的，而加权的重要性抽样是有偏的 (尽管偏差渐近收敛到零)。另一方面，普通重要性抽样的方差一般是无界的，因为比率的方差可以是无界的，而在加权估计值中，任何单一收益的最大权重是 1。事实上，假设有界回报，即使比率本身的方差为无穷大 (Precup, Sutton, Dasgupta 2001)，加权重要抽样估计值的方差也收敛到零。在实践中，加权估计值通常具有较低的方差，并且是较强的首选。然而，我们不会完全放弃普通的重要性抽样，因为它更容易扩展到使用函数逼近的近似方法，我们将在本书的第二部分中探索。普通抽样和加权重要性抽样的每一次抽样方法都是有偏差的，但是，随着样本数量的增加，偏差会逐渐趋于零。在实践中，每次访问方法通常是首选的，因为它们消除了跟踪访问了哪些州的需要，而且它们更容易扩展到近似。在 110 页的下一节中，给出了一种完整的每一次访问 MC 算法，用于使用加权重要性抽样进行离策略评估。

练习 5.5 考虑一个 MDP 和一个非终结符状态和一个行动转换回非终结符状态概率 p 和转换到终端状态的概率 1−p。在所有转换让奖励 + 1, 并让 = 1。假设你观察到一个情节持续 10 步，然后返回 10 步。非终端状态的价值的第一次访问和每一次访问估计是什么??

示例 5.4: 对 Blackjack 状态值的离策略估计，我们使用了普通的和加权的输入-抽样方法，从离策略数据中估计单个 Blackjack 状态的值 (示例 5.1)。回想一下蒙特卡罗方法的优点之一是，它们可以用来评估单个状态，而不需要对其他任何状态进行估计。在这个例子中，我们评估了发牌人展示一张牌的状态，玩家牌的总数是 13 张，玩家有一张可用的 ace(即玩家持有一张 ace 和一张 deuce，或相当于 3 张 ace)。数据是从这个状态开始生成的，然后选择按相同概率 (行为策略) 随机命中或随机选择。目标策略是只坚持 20 或 21，如示例 5.1 所示。这种状态下的价值目标政策大约是 −0.27726(这是由分别生成一百集使用目标政策和平均回报)。这两种离策略方法在使用随机策略的 1000 次离策略事件后都接近这个值。为了确保他们能可靠地做到这一点，我们进行了 100 次独立的实验，每一次实验都从零开始，学习了 10000 集。图 5.3 显示了合成的学习曲线——每个方法的估计的平方误差作为周期数的函数，在 100 次运行中平均。两种算法的误差接近于零，但加权输入-抽样方法在开始时误差小得多，这在实践中是很典型的。

图 5.3: 加权重要性抽样产生的错误估计值较低，即从策略外的事件中得到的单个 21 点状态的值。

例 5.5: 无限大方差一般重要性抽样的估计值具有无限大的方差，因此，当规模收益有无限大的方差时，其收敛性是不令人满意的。图 5.4 中显示了一个简单的示例。只有一个非终结态 s 和两个动作，右和左。右边的动作导致确定性的过渡到终止，而左边的动作，概率是 0.9，回到 s，概率是 0.1，到终止。后一种转变的回报是 +1，否则为 0。考虑总是选择左边的目标策略。在此策略下的所有阶段都包含一定数量的转换 (可能为零)

到 s 后终止，奖励和回报 +1。因此年代的价值目标政策下是 1(= 1)。假设我们估计这个值从 off-policy 数据使用的行为策略, 选择正确的, 剩下的概率相等。

10 万 1 千 1 百万 1 千 1 百万 1 万

集 (对数尺度)

图 5.4: 普通重要性抽样在 inset 显示的单状态 MDP 上产生了惊人的不稳定估计 (示例 5.5)。正确估计这里是 1(= 1), 而且, 即使这是一个示例返回的期望值 (重要性抽样后), 样本的方差是无限的, 估计不收敛于这个值。这些结果是针对非政策的首次访问 MC。

图 5.4 的下半部分显示了使用普通重要性抽样的第一次访问 MC 算法的 10 次独立运行。即使在经历了数百万次的事件之后，估计也没有达到 1 的正确值。相比之下，加权重要采样算法会在第一集以左动作结束后给出准确的 1 的估计值。所有收益不等于 1(即以正确的行动) 是不一致的与目标政策, 因此将会有一个 t:T(T)−1 的零, 既有助于 (5.6) 的分子和分母。加权重要性-抽样算法只产生与目标策略一致的回报率的加权平均数，所有这些都是 1。我们可以通过一个简单的计算来验证在这个例子中，输入- samplingscale 返回的方差是无穷大的。方差的随机变量 X 的期望值是偏离其平均 X, 可以写

Var[X]。= E

−X 2 吗? = E

X2−2 xx̄+ X2
= E
X2
−X2。
因此，如果均值是有限的，在我们的例子中，当且仅当
随机变量的平方的期望是无限的。因此，我们只需要证明，重要性抽样规模收益的期望平方是无限的:
海尔哥哥 吗?T−1 ?t = 0 (圣)| 在 | b(St)G0 2
为了计算这个期望，我们将它分解为基于事件长度和终止的情况。首先要注意，对于任何以正确动作结尾的情节，重要性抽样比率为零，因为目标策略永远不会采取这种行动; 因此，这些事件对期望没有任何贡献 (括号中的数量将为零)，可以忽略。我们只需要考虑包含一些向左动作 (可能为零) 的事件，这些左动作会转换回非终结状态，然后是向左动作转换到终止状态。所有这些事件的返回值都是 1，因此可以忽略 G0 因子。为了得到期望的平方，我们只需要考虑每一集的长度，将每一集发生的概率乘以重要抽样比率的平方，然后相加:
= 0.1 ∞ 吗?k = 0 0.9k • 2k • 2 = 0.2 ∞ 吗?k = 0 1.8 k =∞。
练习 5.6 对于作用值 Q(s, a) 而不是状态值 V (s)，再次给出使用 b 生成的返回值? ?
在学习曲线中，如图 5.3 中所示，练习 5.7 通常会随着训练而减少，就像普通的重要性抽样方法一样。但对于加权输入-抽样方法误差先增大后减小。为什么你认为这是真的吗? ?
练习 5.8 结果为例 5.5，如图 5.4 所示，使用了第一次访问 MC 方法。假设在相同的问题上使用了一个全访问 MC 方法。估计量的方差仍然是无穷大吗? 为什么或为什么不呢??

5.3 增量实现

蒙特卡罗预测方法可以逐步实现，每一集都可以使用第 2 章 (2.4 节) 所述技术的扩展。而在第二章中我们平均回报，在蒙特卡罗方法中我们平均回报。在所有其他方面，与第 2 章中使用的方法完全相同，可以用于 on-policy Monte Carlo 方法。对于非策略蒙特卡罗方法，我们需要分别考虑使用普通重要性抽样的方法和使用加权重要性抽样的方法。在普通的重要性抽样, 返回相应的重要性抽样比率 t:T(T)−1(5.3), 然后简单平均, 如 (5.5)。对于这些方法，我们可以再次使用第 2 章的增量方法，但是使用按比例计算的回报来代替那一章的回报。这就剩下了使用加权重要性抽样的非策略方法。这里，我们必须形成一个加权平均的回报，并且需要一个稍微不同的增量算法。假设我们有一个返回序列 G1 G2···,Gn−1, 开始在同一个国家和每一个都有相应的随机 Wi 重量 (例如,Wi = ti:T(ti)−1)。我们希望作出估计
并保持它的最新，因为我们获得一个额外的回报 Gn。除了跟踪 Vn 之外，我们还必须为每个状态维护给定的前 n 个返回的权重的累积和 Cn。Vn 的更新规则是和
n + 1。= Cn + Wn + 1,
C0 的地方。= 0 (V1 是任意的，因此不需要指定)。下一页的框包含一个完整的逐集递增算法，用于蒙特卡罗政策评估。off-policy 算法是名义上的情况下, 使用加权重要性抽样, 但也适用于对政策的情况下, 通过选择目标和行为的政策是相同的 (在这种情况下 (= b),W 总是 1)。近似 Q 收敛于 q (所有遇到政府行动对), 而根据一个潜在的不同的政策选择行为,b。练习 5.9 修改第一次访问 MC 策略评估的算法 (第 5.1 节)，以使用第 2.4 节中描述的样本平均值的增量实现。? 练习 5.10 从 (5.7) 推导出加权平均更新规则 (5.8)。遵循无加权规则的推导模式 (2.3)。?

5.4 场外蒙特卡罗控制

我们现在准备展示我们在本书中考虑的第二种学习控制方法的例子: 非策略方法。回想一下，on-policy 方法的显著特征是，它们在使用策略进行控制时估计策略的价值。在非策略方法

中，这两个函数是分开的。用于生成行为的策略 (称为行为策略) 实际上可能与被评估和改进的策略 (称为目标策略) 无关。这种分离的一个优点是目标策略可能是确定性的 (例如，贪心)，而行为策略可以继续对所有可能的操作进行示例。非策略蒙特卡罗控制方法使用前两部分中介绍的技术之一。他们在学习和改进目标政策的同时遵循行为政策。这些技术要求行为策略选择目标策略 (覆盖率) 可能选择的所有操作的概率是非零的。为了探究所有的可能性，我们要求行为策略是软的 (即:，选择所有非零概率状态下的所有动作)。盒子在下一页显示了 off-policy 蒙特卡罗控制方法, 基于谷歌价格指数和加权重要性抽样估计 $*$ 和 q$*$。目标政策 $*$ 贪婪的政策就问, 这是一个估计的 q 。政策的行为可以是任何东西, 但为了保证收敛的的最优政策, 无限的回报必须获得每一对状态和行动。这可以保证通过选择 b -soft。政策 收敛于最优状态即使遇到行动是选择根据不同的软政策 b, 这可能改变之间, 甚至在事件。

一个潜在的问题是，该方法只从事件的尾部学习，而所有事件中剩余的动作都是贪婪的。如果不贪心的行为是普遍的，那么学习将是缓慢的，特别是对于那些出现在长时期早期的国家。这可能会大大降低学习速度。使用非政策蒙特卡罗方法来评估这个问题的严重程度的经验是不够的。如果它是严肃的，最重要的解决方法可能是合并时间差异学习，算法思想在下一章中发展。或者, 如果 小于 1, 那么下一节中所开发的想法也可以显著帮助。

练习 5.11 盒装 off-policy MC 算法控制, 你可能期望 W 更新涉及重要性抽样比率 (| St)b(| St), 但是相反，它包含 1 个 b(在 |St)。为什么这是正确的呢? ? 练习 5.12:Racetrack(编程) 考虑像图 5.5 所示的那样，驾驶一辆赛车在一个拐弯处行驶。你想走得越快越好，但不要跑得太快，以免偏离轨道。在我们简化的赛道上，汽车是在一个离散的网格位置，图中的单元格。速度也是离散的，许多网格单元格水平移动和垂直移动每一步。动作是速度分量的增量。每一个可能改变了 + 1,−1 或 0 在每个步骤中, 总共 9(3×3) 行动。两个速度分量都被限制为非负的和小于 5 的，它们不能都是零，除非在起跑线上。每一集都是从一个随机选择的开始状态开始，两个速度分量都为零，当赛车越过终点线时结束。奖励是 −1 每一步直到车穿过终点线。如果汽车撞到轨道边界，它被移动到起跑线上的一个随机位置，两个速度分量被减少到零，并且

起跑线起跑线

图 5.5: 赛道任务的右转数。

事件仍在继续。在每一步更新汽车的位置之前，检查汽车的预定路径是否与轨道边界相交。如果它与终点线相交，那一集就结束了; 如果它与其他地方相交，那么汽车被认为已经到达了轨道边界并被送回起跑线。为了使任务更具挑战性，每一步的概率为 0.1，速度增量都为 0，与预期增量无关。将蒙特卡罗控制方法应用于该任务，从每个起始状态计算最优策略。沿着最优政策 (但关闭这些轨迹的噪声)。?

5.5 Discounting-aware 重要性抽样

到目前为止，我们所考虑的场外政策方法是基于对被认为是单一整体的回报形成重要的抽样权重，而不考虑回报的内部结构作为折现回报的总和。我们现在简要地考虑使用这种结构来显著降低非策略估计量的方差的前沿研究思想。例如, 考虑一下这种情况: 集长, 明显小于 1。具体性, 说去年 100 步和 $= 0$。时间 0 的回报将是 G0 = R1，但它的重要性抽样比率将是 100 年因素, (A0 | S0)b(A0 | S0) (A1 | S1) b(A1 | S1) ••• (A99 | S99) b(A99 | S99)。在普通重要性抽样中，返回将被缩放整个产品, 但它只需要规模第一因子, 通过 (A0 | S0)b(A0 | S0)。其他 99 个因素 (A1 | S1) b(A1 | S1) ••• (A99 | S99) b(A99 | S99) 因为无关在第一次奖励之后，回报已经确定。这些后面的因素都与回报和期望值 1 无关; 它们不会改变预期的更新，但会极大地增加其方差。在某些情况下，他们甚至可以

方差无限。现在让我们考虑一个避免这种巨大的额外差异的想法。这个概念的本质是把折现看成是确定概率终止或者，等价地，部分终止的程度。对于任何 (0,1), 我们可以把返回 G0 部分终止在一个步骤中, 程度 1− , 产生一个返回的第一个奖励,R1, 部分终止后两个步骤,

程度 (1−) , 生产 R1 + R2 的回报, 等等。后者程度对应终止第二步,1− , 而不是已经终止的第一步, 。终止的程度在第三步是这样 (1−) 2, 2 反映的终止不会发生在前两个步骤。这里的部分收益称为平部分收益: Gt:h。= Rt + 1 + Rt + 2 + ••• + Rh,0 t < h t, 其中，“flat”表示没有贴现，“partial”表示这些返回不会一直延伸到终止，而是在 h 处停止，称为 horizon (T 是该集的终止时间)。传统的全收益 Gt 可以看作是平的部分收益之和，如下所示: Gt。= Rt + 1 + Rt + 2 + 2Rt + 3 + ••• + T−−1 t Rt =(1−)Rt + 1 +(1−) (Rt + 1 + Rt + 2) +(1−) 2(Rt + 1 + Rt + 2 + Rt + 3)。

t +(1−) T−−2(Rt + 1 + Rt + 2 + ••• + Rt−1)+ T t−−1(Rt + 1 + Rt + 2 + ••• + Rt)

=(1−) T−1 ?h = t + 1 t h−−1 ḡt:h + T t−−1 ḡt:t。

现在我们需要用同样被截断的重要性抽样比率来衡量平坦的部分回报。Gt:h 只涉及奖励地平线 h, 我们只需要概率之比 h。我们定义一个普通的重要性抽样估计量, 类似于 (5.5),

V(s)。= t (s)

(1−)?T(T)−1 h = T + 1 h T−−1 t:h−1 ḡT:h + T T(T)−−1 t:T(T)−1 ḡT:T(T)

| | T(年代) , (5.9) 与 (5.6) 相似的加权输入-抽样估计量为

V(s)。=

t (s)

(1−)?T(T)−1 h = T + 1 h T−−1 t:h−1 ḡT:h + T T(T)−−1 t:T(T)−1 ḡT:T(T)

t (s)

(1−)?T(T)−1 h = T + 1 h T−−1 t:h−1 + T T(T)−−1 t:T(T)−1 吗?。(5.10) 我们把这两个估计量称为感知重要性的抽样估计量。他们考虑贴现率, 但没有影响 (是一样的 off-policy 估计从 5.5 节) 如果 = 1。

5.6 每个决策重要性抽样

还有一个方法返回的结构作为一笔奖励可以考虑在 off-policy 重要性抽样, 这样可以减少方差即使没有折扣 (即即使 = 1)。在 off-policy 估计量 (5.5) 和 (5.6), 每一项的总和分子本身就是一笔: t:T−1 gt = t:T−1

Rt + 1 + Rt + 2 + ••• + T−−1 t Rt

= t:T rt + 1 + t−1:T−1 rt + 2 + ••• + T T−−1 t:T−1 rt。(5.11) 非策略估计依赖于这些术语的预期值，可以用更简单的方法编写。注意，每个子项 (5.11) 是随机奖励和随机输入-抽样比的乘积。例如，可以使用 (5.3)as 编写第一个子项

t:T rt + 1 =−1 (圣)| b(圣)| (+ 1 | 圣 + 1)b(+ 1 | 圣 + 1) (+ 2 | St + 2) b(+ 2 | St + 2) ••• (圣 −−1 | 1) b(圣 −−1 | 1)Rt + 1。(5.12) 在所有这些因素中，人们可能会怀疑只有第一个和最后一个 (奖励) 是相关的; 所有其他的都是在奖励后发生的事件。此外，所有这些其他因素的期望值是 1: 子邮件

(Ak | Sk)b(Ak | Sk)

。=

一个 b(| Sk) (| Sk)b(一个 | Sk)=

一个 (| Sk)= 1。(5.13)

只要再多走几步，我们就会发现，正如人们所怀疑的那样，所有这些其他因素对期望都没有影响，换句话说，就是

E[t:T rt−1 + 1]= E[t: 泰爱泰党 + 1]。(5.14) 如果我们对 (5.11) 的第 k 项重复这个过程，我们得到

E(t:T−1 rt + k)= E[t:T + k−1 rt + k)。它遵循那我们最初的期望项 (5.11) 可以写 E[t T−1 gt):= E

Gt

在哪里

Gt = t: 泰爱泰党 + 1 + t:t rt + 2 + 1 + 2 t:t rt + 3 + 2 + ••• + T t−−1 t:t−1 rt。我们把这种观点称为“按决策重要性抽样”。它紧跟着, 有另一种重要性抽样估计, 无偏相同的期望 (在第一次访问的情况下) 作为 ordinary-importance-sampling 估计量 (5.5), 使用 Gt:

我们可能期望方差更低。加权重要性抽样是否有每个决策版本? 这是不太清楚。到目前为止，我们所知道的所有估计量都是不一致的 (也就是说，它们不收敛于具有无限数据的真实值)。∗5.13 运动中显示的步骤 (5.14)(5.12)。? ∗ 练习 5.14 修改 off-policy 蒙特卡罗算法控制 (第 111 页) 使用截断加权平均估计量的概念 (5.10)。注意，您首先需要将此方程转换为动作值。?

5.7 总结

本章提出的蒙特卡罗方法通过样本集的形式从经验中学习价值函数和最优策略。与 DP 方法相比，这至少给了他们三种优势。首先，它们可以直接从与环境的交互中学习最优行为，没有环境动力学模型。第二，它们可以与仿真或样本模型一起使用。对于许多应用程序来说，即使很难构造出 DP 方法所要求的转换概率的显式模型，也很容易模拟样本集。第三，将蒙特卡罗方法集中在状态的一小部分上是容易和有效的。一个特别感兴趣的区域可以被精确地评估，而不需要精确地评估状态集的其他部分 (我们将在第 8 章中进一步探讨这个问题)。蒙特卡罗方法的第四个优点，我们在后面的书中讨论过，是它们可能较少受到违反马尔可夫财产的伤害。这是因为它们不根据继承国的价值估计来更新其价值估计。换句话说，这是因为它们没有引导。在设计蒙特卡罗控制方法时，我们遵循了第 4 章中介绍的广义策略迭代 (GPI) 的总体模式。GPI 涉及政策评价和政策改进的相互作用过程。蒙特卡罗方法提供了一个可供选择的政策评估过程。它们并不使用模型来计算每个状态的值，而是平均从状态开始的许多返回值。因为一个状态的值是期望回报，这个平均值可以成为这个值的一个很好的近似值。在控制方法中，我们对近似动作-值函数特别感兴趣，因为这些函数可以用来改进策略，而不需要环境的转换动力学模型。蒙特卡罗方法将政策评估和政策改进步骤按章节进行混合，并可按章节进行递增。保持足够的勘探是蒙特卡罗控制方法中的一个问题。仅仅选择当前估计为最佳的操作是不够的，因为这样就不会获得替代操作的回报，而且可能永远也不会知道它们实际上更好。一种方法是忽略这个问题，假设这些事件从随机选择的状态-动作对开始，以涵盖所有可能性。这种探索的开始有时可以安排在模拟事件的应用程序中，但不太可能从真实经验中学习。在策略方法中，代理总是提交。

探索并尝试找到仍在探索的最佳策略。在 off-policy 方法中，代理也进行了探索，但是学习了可能与所遵循的策略无关的确定性最优策略。策略外预测是指从不同行为策略生成的数据中学习目标策略的值函数。这种学习方法是基于某种形式的重要性抽样，即以两种策略下观察到的行为的概率之比来加权收益，从而将它们的期望从行为策略转换为目标策略。普通重要性抽样使用加权回报率的简单平均数，而加权重要性抽样使用加权平均数。普通重要性抽样产生的是无偏估计，但有较大的，可能是无限的，方差，而加权重要性抽样总是有有限的方差，在实践中是首选的。尽管它们的概念很简单，但用于预测和控制的非策略蒙特卡罗方法仍然不稳定，并且是一个正在进行的研究课题。本章所处理的蒙特卡罗方法与前一章中处理的 DP 方法有两大不同。首先，它们基于样本经验进行操作，因此可以在没有模型的情况下进行直接学习。其次，它们不引导。也就是说，他们不根据其他价值估计来更新他们的价值估计。这两个差异并不是紧密相连的，而且可以分开。在下一章中，我们将考虑从经验中学习的方法，如蒙特卡罗方法，也将考虑引导方法，如 DP 方法。

书目的和历史的言论

“蒙特卡洛”这个词可以追溯到 20 世纪 40 年代，当时洛斯阿拉莫斯的物理学家们发明了一种可能性游戏，他们可以通过研究来帮助理解与原子弹有关的复杂物理现象。在这种意义上对蒙特卡罗方法的报道可以在几本教科书中找到 (例如，Kalos 和 Whitlock, 1986; 鲁宾斯

坦,1981)。

5.1-2 Singh 和 Sutton(1996) 区分了每次访问和第一次访问的 MC 方法和证明了这些方法与强化学习算法的关系。blackjack 示例基于 Widrow、Gupta 和 Maitra(1973) 使用的示例。肥皂泡的例子是一个经典的狄利克雷问题，它的蒙特卡罗解最早由 Kakutani(1945) 提出; 参见赫什和格里戈，1969; 柯南道尔和斯奈尔,1984)。Barto 和 Duff(1994) 讨论了在求解线性方程组的经典蒙特卡罗算法的背景下的政策评估。他们利用 Curtiss(1954) 的分析指出了蒙特卡罗政策评价对大问题的计算优势。

5.3-4 蒙特卡洛在 1998 年出版的这本书中被介绍。可能是蒙特卡罗估计和基于策略迭代的控制方法之间的第一个明确的联系。早期使用蒙特卡罗方法来估计增强学习环境中的动作值是由 Michie 和

6. 第六章 Temporal-Difference 学习

如果要强化学习，就必须把一个想法确定为中心和新颖的，那无疑就是时间性差异 *(TD)* 学习。*TD* 学习是蒙特卡罗思想和动态规划思想的结合。像蒙特卡罗方法一样，*TD* 方法可以直接从原始经验中学习，而不需要环境动力学模型。像 *DP* 一样，*TD* 方法更新估计部分基于其他学习的估计，而不需要等待最终结果 *(*它们是自启动的*)*。在强化学习理论中，*TD*、*DP* 和蒙特卡罗方法之间的关系是一个反复出现的主题*;* 这一章是我们对它的探索的开始。在我们完成之前，我们将看到这些想法和方法相互融合，可以以多种方式组合在一起。特别是，在第七章我们介绍 *n-step* 算法，提供一个桥梁从 *TD* 蒙特卡罗方法，在第十二章我们介绍了 *TD( )* 算法，它无缝地结合。像往常一样，我们首先关注政策评估或预测问题，估计价值函数的问题 v 为给定的政策。对于控制问题 *(*寻找最优策略*)*，*DP*、*TD* 和蒙特卡罗方法都使用广义策略迭代 *(GPI)* 的一些变化。方法上的差异主要是预测问题方法上的差异。

6.1 TD 预测

TD 和蒙特卡罗方法都利用经验来解决预测问题。给出一些经验后政策 ，两种方法更新他们的估计 V v 的非终结符州圣发生在体验。粗略地说，蒙特卡罗方法要等到访问结束后返回，然后使用该返回作为 V (St) 的目标。一种适用于非平稳环境的简单的 every-visit 蒙特卡罗方法。V(St)←(St)+ Gt−V(St) , (6.1) Gt 是实际返回时间 t 后，和 是一个恒定的步长参数 (出口的。, 方程 2.4)。让我们调用这个方法 constant- MC。而蒙特卡罗方法必须等到结束的事件来确定增量 V(St)(只有 Gt),TD 方法需要等待直到下一个时间步。在 t+1 时刻，他们立即形成一个目标，利用观察到的奖励 Rt+1 和估计值 V (St+1) 进行有用的更新。最简单的 TD 方法进行更新

V(St)←(St)+
Rt + 1 + V(圣 + 1)−V(St)
(6.2)

立即过渡到 St+1 并接收 Rt+1。实际上，蒙特卡洛更新 Gt 的目标，而目标 TD 更新 Rt + 1 + V(圣 + 1)。这个 TD 方法叫做 TD(0)，或一步 TD，因为它是一个特例的 TD( ) 和 n-step TD 方法开发的第十二章和第七章。下面的框以过程的形式指定 TD(0)。

因为 TD(0) 的更新部分基于现有的估计，所以我们说它是一种引导方法，就像 DP 一样。从第三章我们知道

v (年代)。= E [Gt | 圣 = s] (6.3) = E [Rt + 1 + Gt + 1 | 圣 = s] (从 (3.9)) = E [Rt + 1 + v (圣 + 1)| 圣 = s]。(6.4) 粗略地说，蒙特卡罗方法以 (6.3) 估计值为目标，而 DP 方法以

(6.4) 估计值为目标。蒙特卡罗目标是一个估计，因为 (6.3) 中的期望值未知; 示例返回用于替代实际期望返回。DP 的目标是估计不是因为预期值, 这是假定为完全的模型所提供的环境, 而是因为 v (圣 + 1) 是未知的和当前估计,V(圣 + 1), 而不是使用。TD 的目标是估计有两个原因: 它样品预期值 (6.4), 它使用当前估计代替真正的 v V。因此，TD 方法结合了采样。

蒙特卡洛与 DP 的 bootstrapping。正如我们将看到的，只要小心和想象，这将使我们在获得蒙特卡罗和 DP 方法的优点方面取得很大的进展。TD(0) 右边显示的是表格式 TD(0) 的备份图。在备份图顶部的状态节点的值估计是基于从它到下一个状态的一个示例转换而更新的。我们称 TD 和蒙特卡罗更新为样本更新, 因为它们涉及展望一个样本的继任者状态 (或政府行动对), 使用的继任者的价值和奖励来计算一个备份值, 然后更新初始状态的值 (或状态-动作对)。示例更新与预期的更新不同在 DP 方法中，它们基于一个样本继承，而不是所有可能继承的完整分布。最后, 请注意,TD 的数量在括号中 (0) 更新是一种错误, 测量圣的估计价值的区别和更好地估计 Rt + 1 + V(圣 + 1)。这个量，被称为 TD 误差，在强化学习中以各种形式出现: t。= Rt + 1 + V(圣 + 1)−V(St)。(6.5) 请注意，每次的 TD 错误都是当时估计的错误。因为 TD 错误依赖于下一个状态和下一个奖励，实际上直到一个时间步骤之后它才可用。即 t V(St) 的错误, 可以在时间 t + 1。还要注意，如果数组 V 在这一集中没有变化 (在蒙特卡罗方法中没有变化)，那么蒙特卡罗误差可以写成 TD 误差之和: Gt−V(St)= Rt + 1 + Gt + 1−V(St)+ V(圣 + 1)− V(圣 + 1)(从 (3.9)) = t + Gt 1−V(圣 + 1)? = t + t + 1 + 2 Gt + 2−V(圣 + 2)? = t + t + 1 + 2 t + 2 + ••• + T t−−1 t−1 + T−t

GT−V(ST)? = t + t + 1 + 2 t + 2 + ••• + T t−−1 t−1 + T−t

−0

k−t k。(6.6)

如果 V 在事件期间更新 (如 TD(0)))，则该标识并不准确，但是如果步骤大小很小，那么它仍然可以保持大约。这种同一性的归纳在时变学习的理论和算法中起着重要的作用。练习 6.1 如果 V 在发作过程中发生变化，那么 (6.6) 只能保持大约; 两者之间的区别是什么? 让 Vt 表示在 TD 错误 (6.5) 和 TD 更新 (6.2) 中所使用的状态值的数组。重新进行上面的推导，以确定必须添加到 TD 错误总和中的额外金额为了等于蒙特卡罗误差。?

例 6.1: 每天下班开车回家，你试着预测回家需要多长时间。当你离开办公室时，你会注意到时间、星期、天气以及其他可能相关的事情。假设这个星期五你正好 6 点离开，你估计要花 30 分钟才能到家。当你到达你的车时，已经是 6:05 了，你注意到开始下雨了。在下雨的时候，交通通常会比较慢，所以你要重新估计，从那时开始需要 35 分钟，或者总共需要 40 分钟。15 分钟后，你已经按时完成了高速公路的部分。当你走到第二条路的时候，你会把估计的总旅行时间缩短到 35 分钟。不幸的是，此时你被卡在一辆缓慢的卡车后面，道路太窄，无法通过。你最终不得不跟着卡车走，直到你在 6 点 40 分拐到你住的那条小街。三分钟后你就到家了。状态、时间和预测的顺序如下:

国离局，周五 6 点到车，下出公路 2ndary road，后面卡车进入家乡街道到达家时间 (分钟)0 5 20 30 40 43 预计时间 30 35 15 10 30 预计总时间 30 40 35 40 43 这个例子的回报是旅途中每一段经过的时间。1 我们不是打折 (= 1), 从而换取每个状态是实际时间从状态。每个状态的值是期望的时间。数字的第二列给出遇到的每个状态的当前估计值。观察蒙特卡罗方法的操作的一种简单方法是在序列上绘制预测的总时间 (最后一列)，如图 6.1(左) 所示。红色箭头显示推荐的变化预测 constant- MC 方法 (6.1), = 1。这些正是每个状态的估计值 (预计时间) 与实际回报 (实际时间) 之间的误差。例如，当你从高速公路上下来的时候，你以为回家只需要 15 分钟，但实际上却需要 23 分钟。公式 6.1 适用于这一点，并确定出高速公路后的时间估计增量。错误,Gt−V(St), 在这个时间是 8 分钟。假设步长参数 , 是 1/2。然后，在这一经验的结果下，预计的离开高速公路的时间将被向上修正 4 分钟。这个变化可能太大了; 那辆卡车很可能只是一次不幸的事故。无论如何，更改只能在离线状态下进行，也就是说，在您到达家之后。只有在这个时候你才知道实际的回报。在开始学习之前，是否有必要等到最后的结果出来? 假设有一天你再次估计离开办公室后开车回家需要 30 分钟，然后你陷入了严重的交通堵塞。离开办公室 25 分钟后，你仍在高速公路上颠簸前行。你现在

如果这是一个控制问题，目的是减少旅行时间，那么我们当然会这样做让奖励成为过去时间的负值。但因为我们只关心这里预测 (政策评估)，我们可以用正数来保持简单。

情况情况

图 6.1: 蒙特卡罗方法在驾驶回家示例中建议的更改 (左) 和 TD 方法 (右)。

估计回家还需要 25 分钟，总共需要 50 分钟。当你在堵车时，你已经知道你最初估计的 30 分钟太过乐观了。你一定要等到回家后才增加初始状态的估计值吗? 根据蒙特卡罗的方法，你必须这么做，因为你还不知道真正的回报。另一方面，根据 TD 方法，你会立即学到东西，把最初的估计时间从 30 分钟改为 50 分钟。事实上，每一个估计都会被转移到紧随其后的估计。回到我们的第一天开车, 图 6.1(右) 显示的变化预测推荐的 TD 规则 (6.2)(这些是由规则的变化如果 = 1)。每个误差成正比的随时间变化的预测, 预测的时间差异。除了让您在等待流量时有一些事情要做之外，还有几个计算上的原因可以解释为什么基于当前的预测而不是等到知道实际的回报时才终止学习是有益的。我们将在下一节中简要讨论其中的一些问题。

练习 6.2 这是一个练习，可以帮助你对为什么 TD 方法比蒙特卡罗方法更有效的直觉。考虑驾车回家的例子，以及 TD 和蒙特卡罗方法如何解决这个问题。你能想象一种情况:TD 更新比蒙特卡罗更新平均要好吗? 给出一个示例场景——对过去经验的描述和当前状态的描述，您将期望 TD 更新更好。这里有一个提示: 假设你有很多下班开车回家的经验。然后你搬到一个新的建筑和一个新的停车场 (但你仍然在同一地点进入高速公路)。现在你开始学习对新建筑的预测了。你能明白为什么 TD 更新在这种情况下会更好吗? 在原著中也可能发生同样的事情场景吗? ?

124 年第六章:Temporal-Difference 学习

6.2 TD 预测方法的优点

TD 方法更新他们的估计部分基于其他估计。他们从猜测中学习猜测——他们自己引导。这是一件好事吗? 与蒙特卡罗和 DP 方法相比，TD 方法有什么优势? 开发和回答这些问题将占用本书的其余部分以及更多的时间。在本节中，我们简要地预测了一些答案。显然，TD 方法比 DP 方法有一个优势，因为它们不需要环境模型、奖励模型和下一个状态概率分布模型。与蒙特卡罗方法相比，TD 方法的下一个最明显的优点是它们自然地以一种在线的、完全增量的方式实现。使用蒙特卡罗方法时，必须等到一集结束时，因为只有那时才知道返回值，而使用 TD 方法时，只需要等待一个时间步骤。令人惊讶的是，这往往是一个关键的考虑。有些应用程序有很长的一段时间，所以延迟所有的学习直到这一集的结尾太慢。其他应用程序是持续的任务，根本没有发作。最后，正如我们在前一章中所提到的，一些蒙特卡罗方法必须忽略或忽略实验行为所发生的事件，这会极大地降低学习速度。TD 方法不太容易受到这些问题的影响，因为它们从每次转换中学习，而不管后续采取什么操作。但是 TD 方法是否可靠呢? 当然，在不等待实际结果的情况下，从下一种猜测中学习一种猜测是很方便的，但我们能不能保证收敛到正确的答案呢? 幸运的是，答案是肯定的。对于任何固定政策 ,TD(0) 已被证明收敛 v , 意味着一个常数的步长参数如果足够小, 以概率 1 如果根据通常的步长参数减少随机近似条件 (2.7)。大多数收敛证明只适用于上述算法的基于表的情况 (6.2)，也有一些适用于一般线性函数逼近的情况。这些结果将在第 9 章的更一般的设置中讨论。如果 TD 和蒙特卡罗方法都渐近地收敛于正确的预测，那么自然的下一个问题就是“谁先到达那里?”换句话说，哪种方法学得更快? 怎样才能更有效地利用有限的数据? 目前，这是一个悬而未决的问题，因为没有人能够从数学上证明一种方法比另一种方法收敛得更快。事实上，我们甚至不知道什么是表达这个问题最合适的正式方式! 然而在实践中,TD 方法通常被发现收敛速度比 constant- MC 方法随机任务, 见例 6.2。

练习 6.3 从随机游动例子的左图中可以看出，第一集的结果只有 V (a) 的变化，这说明了第一集发生了什么? 为什么只有这个状态的估计值? 改变了吗? 它到底改变了多少? ? 练习 6.4 具体成果图的随机漫步的例子所示的值依赖于步长参数, 。你觉得会影响算法的结论是更

好的, 如果是使用广泛的 值吗? 有不同, 固定值 , 要么算法的表现吗明显好于显示? 为什么或为什么不呢? ? ∗6.5 运动图的随机漫步的例子中,TD 的均方根误差方法似乎在下降, 然后起来, 特别是在高 。是什么导致了这种情况? 你认为这种情况经常发生吗近似值函数初始化? ? 练习 6.6 在例子 6.2 中我们说过随机漫步例子的真实值是，2

对于 A 到 e 的状态，描述至少两种不同的方式这些是可以计算出来的。你猜我们实际上用的是什么? 为什么??

6.3 最优的 TD(0)

假设只有有限数量的经验，比如 10 集或 100 个时间步骤。在这种情况下，使用增量学习方法的常见方法是重复地呈现经验，直到方法收敛到一个答案。在给定一个近似值函数 V 的情况下，对于每次访问一个非终端状态的第 t 步，都要计算 (6.1) 或 (6.2) 指定的增量，但值函数只更改一次，以所有增量之和。然后使用新的值函数再次处理所有可用的经验，从而产生新的总体增量，以此类推，直到值函数收敛。我们称此批更新为批更新，因为只有在处理完每批培训数据之后才进行更新。在批处理更新,TD(0) 确定性收敛于一个回答独立于步长参数, , 只要选择 足够小。constant- MC 方法也收敛确定性在相同条件下, 而是一个不同的答案。理解这两个答案将有助于我们理解这两种方法的区别。在正常的更新方法中，这些方法不会一直移动到它们各自的批次答案中，但在某种意义上，它们会在这些方向上采取步骤。在尝试理解这两个答案之前，我们先看几个例子。例 6.3: 随机漫步在批处理更新分批更新版本的 TD(0) 和 constant- MC 如下随机漫步预测例子 (例如 6.2)。在每一集之后，到目前为止看到的所有剧集都被当作一集来处理。他们一再提出的算法,TD(0) 或 constant- MC, 值足够小, 聚合函数。结果值函数然后与 v 相比, 平均根均方误差在 5 个州 (在 100 年独立重复整个实验) 是获得绘制

如图 6.2 所示的学习曲线。注意，批处理 TD 方法始终优于批处理蒙特卡罗方法。

。0 0。。实施率达 1 。2 为

0 25 50 75 100。道明 MC 批处理培训

走/集 RMS 误差，对状态求平均值

图 6.2:TD(0) 的性能和随机漫步 constant- MC 下批培训任务。下批培训,constant- MC 收敛于价值观,V(s), 样本平均数的实际回报有经验在访问每个状态。这些都是最优估计, 他们均方误差最小化训练集的实际回报。在这个意义上令人惊奇的是, 一批 TD 方法能够根据根均方误差表现更好, 右边的图所示。为什么批量 TD 能够比这种最优方法表现得更好? 答案是蒙特卡罗方法只在有限的范围内是最优的。以一种与预测回报更相关的方式，TD 是最优的。示例 6.4: 您现在是一个预测者，您将自己置于一个未知的马尔可夫回报过程的预测者的角色。假设你观察以下 8 集:

一个 0 B,0 B,1 B,1 B,1 B,1 B,1 B,1 B,0

这意味着第一集从状态 A 开始，以 0 的奖励转移到 B，然后以 0 的奖励从 B 结束。其他七集甚至更短，从 B 开始，立即结束。考虑到这批数据，你会说什么是最优预测，估计 V (A) 和 V (B) 的最佳值? 每个人可能都同意，V (B) 的最优值是 3 4，因为在状态 B 中 8 次中有 6 次的进程立即终止，返回 1，而在 B 中的其他 2 次，进程立即终止，返回 0。但是给定这些数据，估计值 V (A) 的最优值是多少? 这里有两个合理的答案。一个是 100

一个 B r = 1

10075

25r = 0

r = 0 乘以进程处于状态 A，它立即遍历到 B(奖励为 0); 因为我们已经确定 B 的值是 3 4，所以 A 必须有值 3 4。看这个答案的一种方法是，它是基于第一次对马尔可夫过程的建模，在这个例子中是向右的，然后计算出给定模型的正确估计，在这个例子中，它给出了 V (A) = 3 4。这是

也就是批次 TD(0) 给出的答案。另一个合理的答案是，我们观察到，我们已经看到了一次，之后的回报是 0; 因此我们估计 V (A) 为 0。这就是批量蒙特卡罗方法给出的答案。注意，这也是对训练数据给出最小平方误差的答案。事实上，它对数据的误差为零。但我们仍然期待第一个答案会更好。如果这个过程是马尔可夫过程，我们预计第一个答案将在未来的数据上产生更低的误差，即使蒙特卡罗的答案在现有数据上更好。示例 6.4 说明了批量 TD(0) 方法和批量蒙特卡罗方法的估计之间的一般差异。批量蒙特卡罗方法总是找到最小化训练集上的均方误差的估计，而批量 TD(0) 方法总是找到对马尔可夫过程的最大似然模型完全正确的估计。一般来说，参数的最大似然估计是生成数据的概率最大的参数值。在这种情况下, 最大似然估计的模型是马尔可夫过程中形成明显的方式从观察到的事件: 转移概率估计从我到 j 是观察到的分数转换从我这去 j, 和相关的预期回报的平均回报观察这些转换。在这个模型中，我们可以计算值函数的估计值，如果模型是完全正确的，那么它将是完全正确的。这被称为确定-等价估计，因为它等价于假定底层过程的估计是确定的而不是被估计的。总的来说，批量 TD(0) 收敛于确定性等价估计。这有助于解释为什么 TD 方法比蒙特卡罗方法收敛得更快。在批处理形式中，TD(0) 比蒙特卡罗方法更快，因为它计算的是真实的确定性-等价估计。这解释了 TD(0) 在随机漫步任务的批处理结果中所显示的优点 (图 6.2)。与某些等价估计的关系也可以在一定程度上解释非批量 TD(0) 的速度优势 (例如，例 6.2，第 125 页，右图)。虽然非批处理方法既没有达到确定的等价性，也没有达到最小的平方误差估计，但是它们可以被理解为大致在这些方向上移动。非批量 TD(0) 可能比 constant- MC 更快, 因为它正朝着一个更好的估计, 即使它是得不到的。目前，对于在线 TD 和蒙特卡罗方法的相对效率，没有什么比这更明确的了。最后，值得注意的是，虽然确定性等价估计在某种意义上是一个最优解，但直接计算它几乎是不可能的。如果 n = |S| 是状态数，则只需按 n2 个内存的顺序形成流程所需的最大似然估计值，按 n3 的计算步骤按顺序计算相应的值函数。在这些术语中，TD 方法可以使用不超过 n 阶的内存和训练集上的重复计算来近似相同的解决方案，这确实令人惊讶。在状态空间较大的任务中，TD 方法可能是近似确定等价解的唯一可行方法。

∗ 锻炼 6.7 设计一个 off-policy TD(0) 版本的更新, 可以使用

任意目标策略 和覆盖的行为策略 b, 在每个步骤使用的重要性抽样比例 t:t(5.3)。?

6.4 Sarsa: On-policy TD Control

现在我们来谈谈 TD 预测方法在控制问题中的应用。像往常一样，我们遵循广义策略迭代 (GPI) 的模式，只是这次使用 TD 方法进行评估或预测部分。与蒙特卡罗方法一样，我们面临的需要是放弃勘探和开发，而方法又分为两大类: 政策上的和政策上的。在本节中，我们介绍一种基于策略的 TD 控制方法。第一步是学习动作-值函数，而不是状态-值函数。特别是, 在政策的方法我们必须估计 q (s) 为当前行为政策 和所有国家年代和行动, 这可以通过使用基本相同的学习 v TD 方法上面所描述的。回想一下，一集由状态的交替序列和状态-动作对组成:

在 Rt + 1 + 1 St Rt 圣 + 1 + 2 + 2 Rt + 3 圣圣 + 3 + 3 + 2 。。

在上一节中，我们考虑了从状态到状态的转换，并学习了状态的值。现在我们考虑从状态-动作对到状态-动作对的转换，并学习状态-动作对的值。在形式上，这些情况是相同的: 它们都是带有奖赏过程的马尔可夫链。保证在 TD(0) 下状态值收敛的定理也适用于相应的作用值算法:

问 (St)←Q(圣,)+

Rt + 1 + Q(圣 + 1,+ 1)−Q(圣,)

。(6.7)

撒尔沙这个更新是在从非终端状态 St. 的每次转换之后完成的，如果 St+1 是终端，那么 Q(St+1，在 +1) 被定义为零。这个规则使用事件的五倍中的每个元素 (St, At, Rt+1, St+1, At+1)，它们构成了从一个状态-动作对到下一个状态-动作对的转换。这个五倍产生了算法的

名称 Sarsa。Sarsa 的备份图如图右侧所示。基于 Sarsa 预测方法设计一种基于策略的控制算法是非常简单的。在所有政策方法, 我们不断地估计 q 行为的政策, 同时改变对贪吃对 q 。Sarsa 控制算法的一般形式在下一页的框中给出。撒尔沙算法的收敛性质取决于政策的依赖的本质问题: 例如, 你可以使用 -greedy 或 -soft 政策。撒尔沙以概率 1 收敛于最优的政策和行为价值函数只要所有政府行动对访问了无限次数和政策收敛极限贪婪的政策 (例如, 可以安排 -greedy 政策通过设置 = 1 / t)。练习 6.8 显示的行为价值版本 (6.6) 适用于 TD 的错误的行为价值形式 t = Rt + 1 + Q(圣 + 1,+ 1)−Q(圣,), 再假设值不要一步一步地改变。?

示例 6.5: 下面显示的多风网格世界是一个标准的网格世界，有开始状态和目标状态，但有一个不同之处: 有一股横风向上穿过网格的中部。动作是标准的向上、向下、右、左四种状态，但在中间区域，由于“风”的作用，产生的下一个状态会向上移动，而“风”的强度会随列而变化。风的强度

0 1000 2000 3000 4000 5000 6000 7000 8000 0 50 100 150 170

时间步长 S G

0 0 0 0 1 1 1 1 2 行动在每一列下面给出，表示向上移动的单元格数。例如，如果你是目标右侧的一个单元格，那么左边的操作将你带到目标上方的单元格。这是一个尚未完全情景任务, 不断回报 −1 直到到达目标状态。右边的图显示的结果应用 -greedy 撒尔沙这个任务, = 0.1, = 0.5, 和初始值 Q(,)= 0 年代,a。增加的斜率图表明, 目标是达到更快。通过 8000 个时间步长，贪心策略自最优以来时间较长 (其中显示一条轨迹); 继续 -greedy 探索保持平均集长度约 17 步骤, 两个超过最低 15。注意，蒙特卡罗方法在这里不容易使用，因为不是所有策略都保证终止。如果发现了一项使代理保持相同状态的策略，那么下一次事件将永远不会结束。像 Sarsa 这样的在线学习方法没有这个问题，因为他们在这一事件中很快就学会了这样的政策是很差的，转而去做别的事情。练习 6.9: 有风的网格世界与国王的移动 (编程) 重新解决了有风网格世界，假设有八个可能的动作，包括对角线的移动，而不是

通常四个。额外的行动你能做得更好吗? 你能做得更好吗? 包括第九次行动，除了引起的行动以外没有任何行动风吗? ?

练习 6.10: 随机风 (编程) 用 King 的动作重新解决了有风的 gridworld 任务，假设风的作用 (如果有的话) 是随机的，有时是由每一列的平均值变化 1。也就是说，有三分之一的时间你完全按照这些值移动，就像之前的练习一样，但也有三分之一的时间你移动一个单元格在上面，另外三分之一的时间你移动一个单元格在下面。举个例子，如果你在目标的右边是一个细胞，你向左移动，那么三分之一的时间你移动一个细胞在目标之上，三分之一的时间你移动两个细胞在目标之上，三分之一的时间你移动到目标。?

6.5 Q-learning: Off-policy TD Control

强化学习的早期突破之一是开发一种称为 Q-learning (Watkins, 1989) 的离线 TD 控制算法

问 (St)←Q(圣,)+

Rt + 1 + max 一个 Q(圣 + 1 a)−Q(圣,)

在这种情况下, 学习行为价值函数,Q, 直接接近问 ∗, 最优行为价值函数, 独立的政策被跟踪。这极大地简化了算法的分析，并使早期的收敛证明成为可能。该策略仍然具有影响，因为它决定访问和更新哪些状态操作对。然而，正确收敛所需要的是所有对继续更新。正如我们在第 5 章中看到的，这是一个最小的需求，因为任何保证在一般情况下找到最优行为的方法都必须需要它。这种假设下的一种变体通常随机步长参数序列的近似条件, 问了问 ∗ 收敛概率 1 所示。Q-learning 算法如下程序形式所示。

Q-learning 的备份图是什么? 规则 (6.8) 更新状态-动作对，因此顶部节点 (更新的根) 必须是一个小的、填充的动作节点。更新也来自操作节点，在下一个状态中最大化所有这些操作。因此，备份图的底部节点应该是所有这些操作节点。最后，请记住，我们指出了使用这些“下一个动作”节点的最大值 (图 3.4-右)。你现在能猜到这个图是什么吗? 如果有，请在翻到第

134 页图 6.4 的答案之前做一个猜测。示例 6.6:Cliff Walking 这个 gridworld 示例比较 Sarsa 和 Q-learning，突出显示了 on-policy (Sarsa) 和 off-policy (Q-learning) 方法之间的区别。

年代 G

theclff R R = 1

更安全的路径

最优路径

R = -100 考虑向右显示的网格世界。这是一项标准的不间断的间歇性任务，有开始和目标状态，以及导致向上、向下、右和左移动的通常动作。奖励是 −1 在所有转换进入该地区除标有“悬崖。“走进这个地区增加了一个奖励 100− 并发送代理立即回到开始。

撒尔沙

q 学习的情节期间的奖励总和

-100 年 0 100 200 300 400 500 右边的图显示了撒尔沙和 q 学习方法的性能与 -greedy 行动选择, = 0.1。在初始的瞬变后，Q-learning 学习为最优策略学习值，即沿着悬崖边缘移动的策略。不幸的是, 这将导致其偶尔因为 -greedy 跌落悬崖的行动选择。另一方面，Sarsa 考虑了动作选择，通过网格的上部学习更长的更安全的路径。虽然 Q-learning 实际上学习了最优策略的值，但是它的在线性能比 Sarsa 差学会迂回的政策。当然, 如果 是逐渐减少, 那么这两种方法将渐近收敛于最优政策。练习 6.11 为什么 Q-learning 被认为是一种策略外的控制方法? 吗? 假设行动选择是贪婪的。Q-learning 算法和 Sarsa 算法完全一样吗? 他们会做出完全相同的动作选择和重量吗更新? ?

6.6 预期撒尔沙

考虑一下学习算法，它就像 Q-learning 一样，只不过它不是使用下一个状态-动作对的最大值，而是考虑到在当前策略下每个动作的可能性。也就是说，考虑使用更新规则的算法。

问 (St)←Q(圣,)+

Rt + 1 + E [Q(圣 + 1,+ 1)| 圣 + 1]−Q(圣,)

←Q(St)+

Rt + 1 +

一个 (| 圣 + 1)Q(圣 + 1 a)−Q(圣,)

(6.9)

但那是遵循 Q-learning 模式的。给定下一个状态 St+1，该算法与 Sarsa 在期望中的移动方向一致，因此被称为 expect Sarsa。它的备份图显示在图 6.4 的右边。预期的 Sarsa 在计算上比 Sarsa 更复杂，但反过来，它消除了随机选择 At+1 带来的方差。在同样的经验量下，我们可能会认为它的表现比萨尔萨略好，事实上它通常是这样的。图 6.3 显示了与 Sarsa 和 Q-learning 相比，带有预期 Sarsa 的 cliff-walking 任务的总结结果。在这个问题上，与 Q-learning 相比，Sarsa 保留了显著的优势。此外，预期的 Sarsa 显示出显著的改善

图 6.3:TD 的临时和渐近性能控制方法 cliff-walking 任务作为 的函数。所有算法与 = 0.1 -greedy 政策使用。渐近表演平均超过 100,000 集，而临时表演平均超过前 100 集。这些数据分别为中期和渐进情况下的 5 万次和 10 次。实心圆表示每种方法的最佳过渡性能。改编自 van Seijen et al.(2009)。

q 学习的预期撒尔沙

图 6.4:Q-learning 和预期 Sarsa 的备份图。

撒尔沙/广泛的步长参数 的值。在 cliff walking 中，状态转换都是确定性的，并且所有的随机性都来自策略。在这种情况下, 预期的撒尔沙可以安全地设置 = 1 没有遭受任何退化的渐近性能, 而撒尔沙只能长期表现良好在 一个较小的值, 在短期业绩很差。在这个和其他例子中，预期 Sarsa 对 Sarsa 有一个一致的经验优势。这些悬崖行走结果预期撒尔沙在政策, 但总的来说它可能使用一个政策目标不同政策 生成行为, 在这种情况下, 它成为 off-policy 算

法。例如, 假设 是贪婪的政策而行为更多的探索性; 然后预期 Sarsa 就是 Q-learning。在这个意义上，期望 Sarsa 包含并推广 Q-learning，同时可靠地改进 Sarsa。除了额外的计算开销很小之外，预期的 Sarsa 可能会完全控制其他两种更知名的 TD 控制算法。

6.7 最大偏差和双学习

到目前为止，我们讨论的所有控制算法都涉及到目标策略的构建的最大化。例如, 在 q 学习的目标政策是贪婪的政策在当前行动的价值观, 这是马克斯, 定义和撒尔沙的政策通常是 -greedy, 也涉及到一个最大化操作。在这些算法中，最大值超过估计值被隐式地用作最大值的估计值，这可能导致显著的正偏差。要知道为什么，考虑一个单独的状态 s，其中有许多行为 a 的真实值 q(s, a) 都为 0，但其估计值 q(s, a) 是不确定的，因此分布在一些高于和一些低于 0 的状态。真实值的最大值是零，但是估计值的最大值是正的，一个正的偏差。我们称之为最大化偏差。例子 6.7: 最大化偏倚示例图 6.5 所示的小 MDP 提供了一个简单的例子，说明了最大化偏差如何影响 TD 控制算法的性能。MDP 有两个非终端状态 A 和 b。事件总是从 A 开始，在两个动作之间进行选择，左和右。正确的操作立即转换到终端状态，并返回 0。左行动过渡到 B, 也奖励的零, 有许多可能的行动导致立即终止所有的奖励来自正态分布方差平均 −0.1 和 1.0。因此, 预期回报任何轨迹从左 −0.1 开始, 因此采取了国家总是一个

图 6.5: 简单情景 MDP 的 Q-learning 和双 Q-learning 的比较 (如图所示) 插图)。Q-learning 最初学习左动作的频率要比正确动作高得多，并且总是比 5 -greedy 行为选择与 = 0.1。相反，双 q 学习基本上不受最大偏差的影响。这些数据平均运行超过 10,000 次。最初的行为价值估计是零。随机 -greedy 行动选择的任何关系被打破。

错误。然而，我们的控制方法可能偏向左，因为极大化偏倚使 B 看起来具有正值。图 6.5 显示了 q 学习 -greedy 行动选择最初学会强烈支持左边行动在这个例子。即使在渐近线,q 学习需要左边行动约 5 有没有避免最大偏差的算法? 首先，考虑一个匪徒案例，在这个案例中，我们对许多行为的每个行为的价值都有嘈杂的估计，这些行为都以每个行为在所有游戏中所获得的奖励的样本平均值来获得。正如我们上面所讨论的，如果我们用估计值的最大值作为真实值的最大值的估计值，就会有一个正的最大化偏差。看待这个问题的一种方法是，它是由于使用相同的样本 (play) 来确定最大化行为和估计它的价值。假设我们把在两集和他们学习使用两个独立的估计, 称之为 Q1(a) 和 (a), 每个估计的真正价值 q(a), 对所有 答: 我们可以使用一个估计, 说 Q1, 确定最大化行动 ∗= argmaxa Q1(a), 另,Q2, 提供的估计价值,Q2(∗)= Q2(argmaxa Q1(a))。这将无偏估计,E[Q2(∗)]= q(∗)。我们还可以重复这一过程，将两个估计值颠倒后，得到第二个无偏差的估计值 Q1(argmaxa Q2(a))。这是双重学习的概念。注意，虽然我们学习了两个估计值，但是每个游戏只更新一个估计值; 重复学习可以使内存需求增加一倍，但不会增加每一步的计算量。双学习的想法自然地扩展到实现完整 MDPs 的算法。例如，类似于 Q-learning 的双学习算法，称为双 q 学习，将时间步分为两步，可能是在每一步上抛硬币。如果硬币朝上，

如果硬币是反面，那么同样的更新是在交换了 Q1 和 Q2 之后完成的，所以 Q2 被更新了。这两个近似函数是完全对称的。行为策略可以同时使用动作价值评估。例如, 一个 -greedy 政策双 q 学习可以根据平均 (或总和) 两个行为价值的估计。在下面的框中给出了一个完整的双 q 学习算法。这是用于生成图 6.5 中的结果的算法。在这个例子中，重复学习似乎消除了最大化偏差带来的危害。当然也有双重版本的莎莎和预期的莎莎。

∗ 锻炼 6.13 的更新方程是什么预期撒尔沙的两倍 -greedy 目标政策? ?

6.8 游戏、后州和其他特殊情况

在这本书中，我们试图提出一种统一的方法来处理广泛的任务，但是当然，总是有一些特殊的任务，以专门的方式得到更好的处理。例如，我们的一般方法包括学习一个动作-值函数，但

是在第一章中，我们提出了一个学习玩井字游戏的 TD 方法，它学习了一些更像状态-值函数的东西。如果我们仔细看一下这个例子，就会发现这个函数在通常意义上既没有动作值函数也没有状态值函数。常规的状态值函数评估代理可以选择操作的状态，但是使用的是状态值函数

在代理人采取行动后，井字游戏会评估董事会的位置。让我们调用这些后态和值函数，这些后态值函数。当我们了解了环境动力学的初始部分，但不一定是完整的动力学时，后状态是有用的。例如，在游戏中，我们通常知道动作的直接影响。我们知道每一个可能的棋局的棋局将会是什么位置，但不知道我们的对手会如何回答。后态价值函数是利用这种知识的一种自然方法，从而产生一种更有效的学习方法。传统的动作值函数会从位置映射到值的估计值。但是许多位置移动对产生相同的结果位置，如下面的例子所示:

在这种情况下，位置移动对是不同的，但是产生相同的“后位置”，因此必须具有相同的值。传统的动作价值函数必须分别评估这两对，而后态价值函数将立即同时评估这两对。任何关于左边位置移动对的了解都会立即转移到右边。后状态出现在许多任务中，而不仅仅是游戏。例如，在排队任务中有一些操作，如将客户分配到服务器、拒绝客户或丢弃信息。在这种情况下，行动实际上是根据它们的直接影响来定义的，这是众所周知的。描述所有可能的专门化问题和相应的专门化学习算法是不可能的。然而，本书所阐述的原则应该得到广泛的应用。例如，在通用策略迭代方面，afterstate 方法仍然被恰当地描述，策略和 (afterstate) 值函数以基本相同的方式交互。在许多情况下，人们仍然需要在政策上和政策之外的方法之间进行选择，以管理持续探索的需要。

练习 6.14 描述了如何根据后态重新定义 Jack 的租车任务 (例子 4.2)。为什么，就这个具体的任务而言，会有这样一个重组有可能加快收敛速度？？

6.9 总结

在本章中，我们介绍了一种新的学习方法——时变学习 (TD)，并说明了它如何应用于强化学习问题。像往常一样，我们把整个问题分为预测问题和控制问题。TD 方法是蒙特卡罗方法解决预测问题的替代方法。在这两种情况下，对控制问题的扩展都是通过我们从动态规划中抽象出来的广义策略迭代 (GPI) 概念。这是近似策略和值函数的交互方式，它们都朝着最优值移动。组成 GPI 的两个过程之一驱动值函数准确预测当前策略的收益; 这就是预测问题。其他流程驱动政策改善本地 (例如, -greedy) 对当前值函数。当第一个过程建立在经验的基础上时，关于保持充分探索的复杂性就产生了。我们可以根据 TD 的控制方法是采用政策上的还是政策外的方法来处理这一复杂问题，对其进行分类。Sarsa 是一种策略方法，Q-learning 是一种策略外的方法。正如我们在这里所展示的，预期的 Sarsa 也是一种非政策方法。有第三种方法可以将 TD 方法扩展到控制中，我们在这一章中没有包括这种方法，叫做 actor - critics 方法。这些方法在第 13 章中有详细介绍。本章介绍的方法是目前应用最广泛的强化学习方法。这可能是因为它们非常简单: 它们可以在线应用，只需要很少的计算量，就可以通过与环境的交互产生体验; 它们几乎完全可以用单个方程来表示，这些方程可以用小型计算机程序实现。在接下来的几章中，我们对这些算法进行了扩展，使它们变得稍微复杂一些，并且非常强大。所有的新算法都将保留本文介绍的本质: 它们将能够在线处理经验，而计算量相对较少，并且它们将受到 TD 错误的驱动。本章所介绍的 TD 方法的特殊情况应该被正确地称为一步法、表格法、无模型的 TD 方法。在接下来的两章中，我们将它们扩展到 n 步的表单 (与蒙特卡罗方法的链接) 和包含环境模型的表单 (与规划和动态编程的链接)。然后，在书的第二部分，我们将它们扩展到各种形式的函数逼近，而不是表 (连接到深度学习和人工神经网络)。最后，在本章中，我们已经讨论了完全在强化学习问题的背景下的 TD 方法，但是 TD 方法实际上比这更一般。它们是学习对动力系统进行长期预测的一般方法。例如，TD 方法可能与预测财务数据、寿命、选举结果、天气模式、动物行为、发电站需求或客户购买相关。只有将 TD 方法作为纯粹的预测方法进行分析，不依赖于它们在强化学习中的应用，它们的理论性质

才首次得到充分的理解。即便如此，TD 学习方法的其他潜在应用还没有得到广泛的探索。

7. 第七章 n-step 引导

在本章中，我们统一了前面两章提出的蒙特卡罗 (MC) 方法和一步时间差 (TD) 方法。无

论是 MC 方法还是单步 TD 方法都不是最好的。在本章中，我们介绍了 n-step TD 方法，这些方法概括了这两种方法，以便在需要时能够顺利地从一种方法转换到另一种方法，以满足特定任务的需求。n 步方法跨越一个光谱，一端是 MC 方法，另一端是一步 TD 方法。最好的方法往往介于两个极端之间。另一种看待 n 步方法的好处的方式是，它们将您从时间步的专制中解放出来。通过一个步骤的 TD 方法，相同的时间步骤决定了操作被更改的频率以及引导完成的时间间隔。在许多应用程序中，我们希望能够非常快速地更新操作，以考虑任何已经更改的内容，但是如果是在发生重大且可识别的状态更改的一段时间内，bootstrapping 工作得最好。使用一步 TD 方法，这些时间间隔是相同的，因此必须做出妥协。n 步方法允许在多个步骤上进行引导，使我们摆脱了单时间步的专制。n 步方法的思想通常被用于介绍资格跟踪的算法思想 (第 12 章)，它允许同时在多个时间间隔内进行引导。在这里，我们转而考虑 n 步自举的想法，将对 eligibilitytrace 机制的处理推迟到稍后。这允许我们更好地分离问题，在更简单的 n 步设置中尽可能多地处理它们。和往常一样，我们首先考虑预测问题，然后是控制问题。也就是说，我们首先要考虑的是 n 步方法如何能够帮助预测作为一个固定策略的状态函数的回报。在估算 v)。然后我们将思想扩展到行为值和控制方法。

7.1 n-step TD 预测

蒙特卡罗和 TD 方法之间的方法空间是什么? 考虑估算 v 从样本集生成使用 。蒙特卡罗方法根据从该状态观察到的整个奖励序列，对每个状态执行更新，直到事件结束。另一方面，单步 TD 方法的更新只基于下一个奖励，一步后从状态值开始作为剩余奖励的代理。因此，一种中间方法将基于中间奖励数执行更新: 多于一个，但在终止之前少于所有的。例如，两步更新将基于前两步奖励和两步后状态的估计值。类似地，我们可以有三步更新、四步更新等等。图 7.1 显示了备份图的频谱 n-step v 更新, 一步 TD 更新左边和右边的 up-until-termination 蒙特卡罗更新。

互译 TD 以及 TD(0) 2-step 3-step TD n-step TD。∞ 一步一步 TD 和蒙特卡洛

图 7.1:n 步方法的备份图。这些方法构成了光谱范围从单步 TD 方法到蒙特卡罗方法。

使用 n 步更新的方法仍然是 TD 方法，因为它们仍然根据先前的估计与后来的估计的不同而改变先前的估计。之后的估计不是一步后的，而是 n 步后的。时间差异超过 n 步的方法称为 n 步 TD 方法。上一章介绍的 TD 方法都使用单步更新，这就是为什么我们称它们为单步 TD 方法。更正式地说，考虑状态 St 的估计值的更新，由于状态奖赏序列 St, Rt+1, St+1, Rt+2···, RT, ST(省略动作) 我们知道在蒙特卡洛更新估计 v (St) 更新的方向

完整的回报: Gt。= Rt + 1 + Rt + 2 + 2Rt + 3 + ••• + T−−1 t Rt, 其中 T 是这一集

的最后一步。让我们将这个量称为更新的目标。而在蒙特卡洛更新目标是回报，一步更新目标是第一个回报加上下一个状态的折现估计值，我们称之为一步回报:

Gt:t + 1。= Rt + 1 + Vt(圣 + 1), Vt 的地方:S→R 估计在时间 t 的 v 就在这里。Gt 的下标:t + 1 表明, 这是一个截断换取时间 t 使用奖励直到时间 t + 1, 与贴现估计 Vt(圣 + 1) 取代其他的条款 Rt + 2 + 2Rt + 3 + ••• + T−−1 t rt 的回报, 正如我们在前一章讨论。我们现在的观点是，这个想法在两个步骤之后和一个步骤之后一样有意义。两步更新的目标是两步返回: Gt:t + 2。= Rt + 1 + Rt + 2 + 2Vt + 1(圣 + 2), 现在 2Vt + 1(圣 + 2) 纠正缺失的条款 2Rt + 3 + 3Rt + 4 + ••• + T−−1 t rt。同样，任意 n 步更新的目标是 n 步返回: Gt:t + n。= Rt + 1 + Rt + 2 + ••• + n−1 Rt + n + nVt + n−1(圣 + n)(7.1) 所有 n,t(n 1 和 0 t < t−n。所有 n-step 回报可以被认为是全部返回近似, 截断后 n 步, 然后纠正其余失踪条款 Vt + n−1(圣 + n)。如果 t + n (如果 n-step 返回延伸到或超过终止), 那么所有缺失的条款视为零, 等于定义和 n-step 返回普通全部返回 (Gt:t + n。如果 t + n t = Gt)。注意，n> 1 的 n 步返回包含了从 t 到 t+ 1 转换时不可用的未来奖励和状态。没有真正的算法可以使用 n-step 返回直到它 Rt + n 和 v + n 计算 −1。第一次出现这种情况是 t+ n，使用 n 步返回的自然状态值学习算法是这样的

Vt + n(St)。= Vt + n−1(St)+ ?Gt:t + n−Vt + n−1(St)?,0 t < t(7.2), 而其他州的值保持不变:Vt + n(s)= Vt + n−1(s), 所有的 s ? = 圣。我们称这个算法为 n 步 TD。注意任何变化都是在每集的前 n−1 步骤。为了弥补这一点，在这一集的结尾、结束后和开始下一集之前都要进行同样多的更新。完整的伪代码在下一页的框中给出。练习 7.1 在第 6 章中，我们注意到如果值估计值不随步骤变化，那么蒙特卡罗误差可以写成 TD 误差之和 (6.6)。显示在 (7.2) 中使用的 n 步错误也可以作为一个和 TD 错误 (同样，如果是值的话估计不会改变) 概括先前的结果。? 练习 7.2(编程) 使用 n 步方法，值估计值会随着步骤的不同而变化，因此使用 TD 错误和的算法 (参见前面的练习)

(7.2) 中的错误位置实际上是一个稍微不同的算法。是更好的算法还是更糟糕的算法? 设计并编写一个小实验来回答这个问题问题经验。? n-step 返回使用价值函数 Vt + n−1 纠正丢失的奖励之外 Rt + n。n-step 返回的一个重要属性是他们的期望是保证更好的估计比 Vt v + n−1, 在一个最坏的状态。也就是最糟糕的错误预期 n-step 返回保证小于或等于 n 乘以最严重的错误在 Vt + n−1:

马克斯 s 吗? 吗?? E [Gt:t + n | 圣 = s]−v (s) 吗? 吗? 吗? n 马克斯 s 吗? 吗?? Vt + n(s)−−1 v (s) 吗? 吗?,(7.3) 对于所有 n 1。这被称为 n 步返回的错误减少属性。由于误差减小的性质，我们可以正式地证明，在适当的技术条件下，所有 n 阶 TD 方法都收敛于正确的预测。因此，n 步 TD 方法形成了一组声音方法，其中一步 TD 方法和蒙特卡罗方法是极端成员。示例 7.1: 在示例 6.2 中描述的 5 状态随机漫步任务中，考虑使用 n-step TD 方法。假设第一集直接从中心状态 C 到右边，经过 D 和 E，然后在右边结束，返回 1。回想一下，所有状态的估计值都是从中间值 V (s) = 0.5 开始的。由于这种经验，一步法只会改变最后一种状态的估计，

V (E) 增加到 1，观察到的返回值。另一方面，两步方法将增加前两种状态的值:V (D) 和 V (E) 都将递增为 1。对于 n >2，一个三步法，或者任何一个 n 步法，会将所有三个访问状态的值都增加到 1，所有的值都是相同的。n 的哪个值更好? 图 7.2 显示了一个简单的实证测试的结果为一个较大的随机游走过程,19 个州, 而不是 5(−1 左边的结果, 所有的值初始化为 0), 我们在本章使用作为一个运行的例子。结果显示 n-step TD 方法 n 和 的值范围。显示的性能测量每个参数设置, 在纵轴上, 是 √(之间的平均平方误差预测最后一集的 19 个州和自己的真实值, 然后平均在第十集,100 年整个实验的重复 (此句组走都是用于所有参数设置)。注意，中间值为 n 的方法效果最好。这说明了将 TD 和蒙特卡罗方法推广到 n 步方法，可能比这两种极端方法中的任何一种都要好。

0.5

平均均方根误差 0.45

超过 19 个州 0.4 和前 10 个州 0.35 事件 0.3

0.25
0.2 0 0.4 0.8 - 0.6 1

图 7.2: 性能 n-step TD 方法作为 的函数, 针对各种 n 值,19 个随机漫步的任务 (例如 7.1)。

练习 7.3 为什么你认为在本章的例子中使用了一个更大的随机行走任务 (19 个州而不是 5 个州)? 走小一点的路会把优势转移到 n 的不同值吗? 左侧的变化结果如何从 0 到 −1 在大走路? 你认为这对 n 的最大值有影响吗??

7.2 n-step 撒尔沙

n 步方法如何不仅用于预测，还用于控制? 在本节中，我们将展示 n 步方法如何以一种简单的方式与 Sarsa 结合

146 年第七章:n-step 引导

生成一种策略 TD 控制方法。我们称 Sarsa 的 n 阶版本为 n 阶 Sarsa，而在上一章中出现的原始版本我们将其命名为 one-step Sarsa(0) 主要的思想是简单的开关状态的行为 (政府行动对), 然后使用一个 -greedy 政策。n 步 Sarsa 的备份图 (如图 7.3 所示) 与 n 步 TD 的备份图 (图 7.1) 一样，是交替状态和动作的字符串，除了 Sarsa 的备份图都以动作而不是状态开始和结束。我们根据估计的行动值重新定义 n 步回报 (更新目标):

Gt:t + n。= Rt + 1 + Rt + 2 + ••• + n−1 Rt + n + nQt + n−1(圣 + n + n),n 1,0 t < t−n, (7.4)

Gt:t + n。如果 t + n t = Gt。自然算法是

Qt + n(St)。= Qt + n−1(St)+ (Gt:t + n n−−Qt + 1(圣,)],0 t < t(7.5), 而其他州的值保持不变:Qt + n(,)= Qt + n−1(,), 为所有的年代, 这样一个年代吗? = 圣或? =。我们称之为 n 步萨尔萨算法。伪代码显示在下一页的框中，图 7.4 给出了一个与单步方法相比，它可以加速学习的例子。

互译撒尔沙 Sarsa(0) 2-step Sarsa 3-step Sarsa n-step Sarsa。∞ 要撒尔沙即蒙特卡洛 n-step 预计撒尔沙

图 7.3: 状态操作值的 n 步方法频谱的备份图。它们的范围从 Sarsa(0) 的一步更新到蒙特卡罗方法的上至终止更新。在这两者之间是 n 步更新，基于 n 步的实际奖励和第 n 个下一个状态-行动对的估计价值，所有的都适当地打折。在最右边是 n 阶期望 Sarsa 的备份图。路线动作值增加一步萨尔萨动作值增加由所述的撒尔沙图 7.4: 使用 n 步方法加速政策学习的 Gridworld 示例。第一个面板显示了代理在单个事件中所采取的路径，以 a 结尾。高回报的位置，由 g 标记。在这个例子中，所有的值最初都是 0，并且都是 0 除了 g 点的正奖励外，奖励为零。另外两个面板上的箭头显示了通过一步和 n 步 Sarsa 方法，哪些行为值因这条路径而增强。一步法只强化动作序列的最后一个动作导致高回报，而 n 阶方法加强了序列的最后 n 个动作，从这一集中学到了很多。

练习 7.4 证明 Sarsa(7.4) 的 n 阶返回可以精确地用一个新的 TD 错误来写，as

Gt:t + n = Qt−1(圣,)+ 最小值 (t + n t)−1 ?k = t k−t[Rk + 1 + Qk(Sk + 1, 正义与发展党 + 1)−Qk−1(Sk,Ak)]。

(7.6)

预期的撒尔沙呢? 预期 Sarsa 的 n 步版本的备份图显示在图 7.3 的最右侧。它由一个样本的线性字符串操作和状态, 就像在 n-step 撒尔沙, 除了它的最后一个元素是一个分支在所有行动可能性加权, 一如既往, 概率在 。这个算法可以用与 n 步 Sarsa(上) 相同的方程来描述，除了 n 阶返回重新定义为。

Gt:t + n。= Rt + 1 + ••• + n−1 Rt + n + n∇t + n−1(圣 + n),t + n < t)(7.7)(Gt:t + n。= Gt t + n t)∇t(s) 是预期的近似值的年代, 使用估计的行动值在时间 t, 根据目标政策:

Vt(年代)。=
一个 (|)Qt(,) 年代。(7.8)
期望近似值用于开发本书其余部分的许多动作-值方法。如果 s 是终端，则其期望值定义为 0。

7.3 n-step Off-policy 学习

回想一下,off-policy 学习是学习一个政策的价值函数, , 而另一项政策后,b。通常, 的贪婪策略目前 action-value-function 估计, 和 b 是一个探索性的政策, 也许 -greedy。为了使用来自 b 的数据，我们必须考虑到两个策略之间的差异，使用它们采取所采取行动的相对概率 (见第 5.5 节)。在 n 步方法中，返回是在 n 步之上构建的，所以我们对这些 n 个动作的相对概率感兴趣。例如，要制作一个简单的 n 步 TD 的脱机版本，只需对时间 t(实际上是在时间 t+ n) 进行加权 t:t + n−1:

Vt + n(St)。= Vt + n−1(St)+ t:t + n−1(Gt:t + n−Vt + n−1(St)],0 t < t(7.9) t:t + n−1, 称为重要性抽样比率, 相对概率的两种策略下的 n 行为从 + n−1(cf Eq。5.3):

例如, 如果任何一个永远不会采取行动的 (即。 (Ak | Sk)= 0) 然后 n-step 返回应给予零重量和完全忽略。另一方面, 如果偶然一个 的行动将有更大概率比 b, 那么这将增加重量, 否则会返回。这是有意义的, 因为操作的特点是 (因此我们想了解它), 但选择 b, 因此很少出现很少的数据。为了弥补这一点，我们必须在这种情况发生时对其进行超重处理。注意，如果两个策略实际上是相同的 (on-policy 的情况)，那么重要性抽样比率总是 1。因此，我们的新更新 (7.9) 概括并可以完全取代我们之前的 n 步 TD 更新。同样，我们之前的 n 步 Sarsa 更新可以被一个简单的 off-policy 表单完全替代:

Qt + n(St)。= Qt + n−1(St)+ t + 1:t + n(Gt:t + n n−−Qt + 1(圣,)],(7.11) 为 0 t < t。注意，这里的重要性抽样比 n 步 TD(7.9) 晚一步开始和结束。这是因为我们正在更新状态-动作对。我们不需要关心我们选择行动的可能性; 既然我们已经选择了它，我们想要充分地从发生的事情中学习，并且只对后续操作进行重要的抽样。完整算法的伪代码如下所示。

非策略版本的 n-step 预期 Sarsa 将使用与上面对 n-step Sarsa 相同的更新，除了重要性抽样比率将减少一个因素。就是上面的方程使用 t + 1:t + n−1 代替 t + 1:t + n, 当然它会使用预期的撒尔沙版本的 n-step 返回 (7.7)。这是因为在预期的 Sarsa 中，所有可能的行为都在最后的状态中被考虑; 实际上服用的那一种没有效果，也不需要纠正。

7.4 * 每一决策方法与对照不同

上一节中介绍的多步离策略方法很简单，概念上也很清楚，但可能不是最有效的方法。一种更复杂的方法将使用每个决策的重要性抽样思想，如第 5.9 节所介绍的。要理解这种方法，首先要注意，普通的 n 步返回 (7.1) 和所有返回一样，都可以递归地编写。对于在视界 h 处结束的 n 步，可以写入 n 步返回

Gt:h = Rt + 1 + Gt + 1:h t < h < t, (7.12) “大酒店”:h。= Vh−1(Sh)。(回想一下，这个返回在时间 h 时使用，之前表示 t + n。) 现在考虑行为策略 b 后的效果是不一样的目标政策 。所有产生的经验, 包括第一个奖励 Rt + 1 和下一个状态圣 + 1 必须由重要性抽样比例加权时间 t, t = (圣)| b(| St)。人们可能会简单地用上面等式的右边来衡量，但我们可以做得更好。假设行动在时间 t 永远不会选择通过 , 所以 t 是零。然后，一个简单的加权将导致 n 步返回为零，这将导致作为目标时的高方差。相反，在这种更复杂的方法中，我们使用另一种策略定义，即 n 步返回结束于 horizon h, as Gt:h。= t(Rt + 1 + Gt + 1:h)+(1− t)Vh−1(St),t < h < t(7.13), 再次 Gh:h。= Vh−1(Sh)。在这种方法中, 如果 t 是零, 那么目标而不是零, 导致估计缩水, 目标是一样的估计, 导致没有变化。重要性抽样比率为零意味着我们应该忽略样本，因此保持估计不变似乎是合适的。第二项，附加项 (7.13) 称为控制变量 (由于不明确的

原因)。注意，控件变量不会更改预期的更新; 重要性抽样比率的期望值为 1(第 5.9 节)，与估计值不相关，因此控制变量的期望值为零。还要注意,off-policy 定义 (7.13) 是一个严格的泛化的早些时候在政策的定义 n-step 返回 (7.1), 这两个相同的政策情况下, t 总是 1。对于传统的 n 阶方法来说，与 (7.13) 结合使用的学习规则是 n 阶 TD update(7.2)，它除了嵌入在返回中的抽样比率外没有显式的重要性抽样比率。练习 7.5 编写策略外状态值预测算法的伪代码上面所描述的。?

对于动作值，n 步返回的 off-policy 定义有点不同，因为第一个动作在重要性抽样中没有作用。第一个动作是学习到的; 在目标政策下，它是否不可能甚至不可能是不重要的，它已经被采取了，现在必须给它的奖励和状态提供全部的单位重量。重要性抽样只适用于其后的行为。首先要注意的是，对于操作，以 horizon h 结尾的 n 步策略返回值，期望形式 (7.7) 可以像在 (7.12) 中一样递归地写，但是对于操作，递归以 Gh:h 结尾。= Vh−1(Sh) 如 (7.8)。带有控件变量的 off-policy 表单是

Gt:h。= Rt + 1 +
t + 1 gt + 1:h + Vh−1(圣 + 1)− t qh−1 + 1(圣 + 1,+ 1)
= Rt + 1 + t + 1
Gt + 1:h Qh−−1(圣 + 1,+ 1)
+ Vh−1(圣 + 1),t < h t。(7.14)

如果 h < T，那么递归以 Gh:h 结束。= Qh−1(Sh 啊), 但是, 如果 h T , 递归结束和 GT−1:h。= RT。合成预测算法 (与 (7.5) 结合后) 类似于预期的 Sarsa。练习 7.6 证明上述方程中的控制变量不改变收益的期望值。? ∗ 练习 7.7 编写的伪代码 off-policy 行为价值预测算法描述上方。特别注意终止条件到达视界或情节结束时的递归。? 练习 7.8 显示了 n 阶跃返回 (7.13) 的一般 (非策略) 版本仍然可以准确、紧凑地写成基于状态的 TD 错误 (6.5) 的总和近似状态值函数不变。? 练习 7.9 重复上述练习，得到偏离策略的 n 步返回 (7.14) 和预期的 Sarsa TD 错误 (公式中括号中的数量)。

练习 7.10(编程) 设计一个小的离策略预测问题，并使用它来表明使用 (7.13) 和 (7.2) 的离策略学习算法更有效与使用 (7.1) 和 (7.9) 的简单算法相比。? 我们在本节、前一节和第 5 章中使用的重要性抽样使我们能够进行良好的偏离策略的学习，但也会导致高方差更新，迫使使用一个小的步长参数，从而导致学习速度变慢。很可能不可避免的是，非政策培训要比政策培训慢——毕竟，数据与正在学习的东西没有多大关系。然而，这些方法也可以改进。控制变量是减少方差的一种方法。另一种方法是根据观察到的方差快速调整步长，如 Autostep 方法 (Mahmood、Sutton、Degris 和 Pilarski, 2012)。另一种有希望的方法是田 (为准备) 将 Karampatziakis 和 Langford(2010) 的不变更新扩展到 TD。Mahmood 的使用技术 (2017); 马哈茂德

Sutton(2015) 也可能是解决方案的一部分。在下一节中，我们将考虑一个不使用重要性抽样的非策略学习方法。

7.5 无重要抽样的脱机学习:n 步树备份算法

在没有重要抽样的情况下，偏离政策的学习是可能的吗?Q-learning 和来自第六章的期望 Sarsa 在单步情况下可以做到这一点，但是有相应的多步算法吗? 在本节中，我们介绍了这样一个 n 步方法，称为树备份算法。

在 Rt + 1 + 1 圣 + 1
圣 + 2 Rt + 2
在 Rt + 3 + 2 + 3

的 3 步 tree-backup 更新该算法的思想由右图所示的三步树备份备份图提出。沿着中心脊柱，在图中标注有三个样本状态和奖励，以及两个样本动作。这些是表示初始状态-动作对 St 之后发生的事件的随机变量。挂在每个状态边的是未被选择的操作。(对于最后一个状态，所

有的动作都被认为还没有被选中。) 因为我们没有未选中操作的示例数据，所以我们引导并使用它们的值估计值来形成更新的目标。这稍微扩展了备份图的概念。到目前为止，我们总是将图顶部节点的估计值更新为目标，并结合沿途的奖励 (适当的折扣) 和底部节点的估计值。在树备份更新中，目标包含所有这些东西，加上悬挂在各个级别上的悬空操作节点的估计值。这就是为什么它被称为树备份更新; 它是对整个估计动作值树的更新。更准确地说，更新是来自树的叶节点的估计操作值。在内部的动作节点，对应于实际采取的行动，不要参与。每个叶节点对目标与体重成正比的概率发生在目标政策 。因此每个一级操作造成的体重 (| 圣 + 1), 除了真正采取行动, 在 + 1, 不贡献。其概率, (+ 1 | 圣 + 1), 用于重量所有二级动作值。因此，每一个非选定的二级动作 a? 贡献与体重 (+ 1 | 圣 + 1) (一个? | 圣 + 2)。每个第三级行动贡献与体重 (+ 1 | 圣 + 1) (圣 + 2)+ 2 | (? ? | 圣 + 3), 等等。它就像图中的一个动作节点的每个箭头都是根据目标策略中被选择的操作的概率来加权的，如果在操作下面有树，那么这个权重就适用于树中的所有叶节点。

我们可以将三步树备份更新看作是由 6 个半步组成的，在一个动作到一个后续状态的半步之间交替进行，并且从这个状态考虑所有可能的动作以及它们在策略下发生的概率。现在让我们为 n 步树备份算法开发详细的方程。一步返回 (目标) 与预期的 Sarsa 相同，

Gt:t + 1。= Rt + 1 +
一个 (圣 + 1)| Qt(圣 + 1), (7.15)
t < t−1, 两步 tree-backup 返回
Gt:t + 2。= Rt + 1 +
在 + 1 ? = (圣 + 1)| Qt + 1(圣 + 1)
在 + 1 | 圣 + + (1)
Rt + 2 +
一个圣 + 2 (|)Qt + 1(圣 + 2)
= Rt + 1 +
在 + 1 ? = (圣 + 1)| Qt + 1(圣 + 1 a)+ (+ 1 | 圣 + 1)Gt + 1:t + 2,
t < t−2。后一种形式给出了树备份 n 步返回的一般递归定义:
Gt:t + n。= Rt + 1 +
在 + 1 ? = (圣 + 1)| Qt + n−1(圣 + 1)+ (+ 1 | 圣 + 1)Gt + 1:t + n(7.16) t < t−1,n 2,n = 1 例由 (7.15) 除了 GT−1:t + n。= RT。然后将此目标与通常的 n 步 Sarsa 动作-值更新规则一起使用:

Qt + n(St)。= Qt + n−1(St)+ (Gt:t + n n−−Qt + 1(圣,),
0 t < t, 而所有其他政府行动对的值保持不变:Qt + n(,)= Qt + n−1(,), 为所有的年代, 这样一个年代吗? = 圣或? =。该算法的伪代码显示在下一页的框中。练习 7.11 表明，如果近似的动作值不变，则树备份返回 (7.16) 可以写成基于预期的 TD 错误之和:

Gt:t+n = Q(St, At) +。最小值 (t + n−1 t−1)?k = t k k ? 我 = t + 1 (Ai | Si),
t 的地方。= Rt + 1 + ∇t(圣 + 1)−Q(圣,) 和 ∇t 由 (7.8) 给出。?

7.6 7.6 * 统一算法:n-step Q( )

到目前为止，在本章中，我们已经考虑了三种不同类型的动作-值算法，对应于图 7.5 所示的前三个备份图。n 阶 Sarsa 有所有的样本跃迁，树备份算法有所有的状态到动作的跃迁完全分支，没有采样，n 阶期望 Sarsa 有所有的样本跃迁，除了最后一个状态到动作的跃迁，它以一个期望值完全分支。这些算法能统一到什么程度? 图 7.5 中的第四个备份图建议了一种统一的思想。这是这样一种想法，即可以一步一步地决定是否要将操作作为示例 (如在 Sarsa 中)，或者考虑对所有操作的期望 (如在树备份更新中)。然后，如果选择总是抽样，就会得到 Sarsa，而如果选择从不抽样，就会得到树备份算法。预期 Sarsa 将是一个案例，其中一个选择除了最后一个步骤之外的所有步骤。

四步撒尔沙四步树备份四步预计撒尔沙四步问 ( )

图 7.5: 本章迄今为止考虑的三种 n 步动作-值更新的备份图 (4 步情况)，加上第四种结合的更新的备份图他们所有人。一半的 表示转换的重要性抽样中需要 off-policy 情况。第四种更新结合其他各州的基础上通过选择是否样品 (t = 1)(t = 0)。

当然还有很多其他的可能性，正如图中最后一张图所示。为了进一步增加可能性，我们可以考虑抽样和期望之间的连续变化。让 t [0,1] 表示取样的程度上一步 t, 用 = 1 表示完整的抽样和 = 0 表示一个纯粹的期望没有抽样。随机变量 t 可能被设定为一个函数的状态, 行动, 或政府行动在时间 t。我们称之为 n-step Q( ) 拟议的新算法。现在让我们开发 n-step Q( ) 的方程。首先我们写 tree-backup n-step 返回 (7.16) 的地平线 h = t + n, 然后在预期的近似值 V(7.8):

Gt:h = Rt + 1 +

在 + 1 ? = Qh− (| 圣 + 1)1(圣 + 1 a)+ (+ 1 | 圣 + 1)Gt + 1:h

= Rt + 1 + Vh−1(圣 + 1)− (+ 1 | 圣 + 1)Qh−1(圣 + 1,+ 1)+ (+ 1 | 圣 + 1)Gt + 1:h = Rt + 1 + (+ 1 | 圣 + 1)

Gt + 1:h Qh−−1(圣 + 1,+ 1)

+ Vh−1(圣 + 1), 之后, 这正是像 n-step 换取撒尔沙与控制变量 (7.14) 除了操作概率 (+ 1 | 圣 + 1) 代替重要性抽样比率 t + 1。问 ( ), 我们这两种情况之间的滑动线性:

Gt:h。= Rt + 1 +

t + 1 t + 1 +(1− t + 1) (+ 1 | 圣 + 1)

Gt + 1:h Qh−−1(圣 + 1,+ 1)

+ Vh−1(圣 + 1), (7.17)

156 年第七章:n-step 引导

t < h t。递归以 Gh:h 结束。= Qh−1(Sh 啊) 如果 h < T, 或与 GT−1:T。如果 h = T。然后我们使用 n-step Sarsa(7.11) 的通用 (off-policy) 更新。在方框中给出了一个完整的算法。

7.7 总结

在本章中，我们开发了一系列时间差异学习方法，它们介于前一章的一步 TD 方法和前一章的蒙特卡罗方法之间。包含中间引导量的方法很重要，因为它们通常比任何一种极端都要好。

四步 TD 本章的重点是 n 步方法，它展望下一个 n 个奖励、状态和行为。右边的两个四步备份图总结了介绍的大多数方法。州值更新显示是 n-step TD 重要性抽样, 和行为价值更新 n-step Q( ), 给出预期撒尔沙和 Q 学习的。所有 n 步方法都需要在更新之前延迟 n 个时间步骤，因为只有这样才能知道所有需要的未来事件。另一个缺点是，与以前的方法相比，它们涉及更多的计算时间步长。与单步方法相比，n 步方法还需要更多的内存来记录最后 n 步的状态、动作、奖励，有时还需要更多的内存。最后，在第 12 章中，我们将看到多步 TD 方法是如何通过使用资格跟踪以最小的内存和计算复杂度实现的，但是除了一步方法之外，总会有一些额外的计算。这样的代价很值得为摆脱单一时间的暴政而付出代价。虽然 n-step 方法比那些使用资格跟踪的方法要复杂得多，但是它们在概念上是清晰的。我们试图利用这一点，在 n-step 案例中开发出两种脱机学习的方法。1、基于重要性抽样概念上很简单，但方差很大。如果目标和行为策略非常不同，那么在实现效率和实用性之前，可能需要一些新的算法思想。另一种是基于树备份更新的 Q-learning 自然扩展到具有随机目标策略的多步情况。它不涉及重要性抽样，但是，如果目标和行为策略本质上不同，即使 n 很大，引导也可能只跨越几个步骤。

书目的和历史的言论

n 阶返回的概念是由 Watkins(1989) 提出的，他也首先讨论了他们的错误减少属性。这本书的第一版探讨了 n 步算法，在这本书中，它们被视为概念性兴趣，但在实践中不可行。Cichosz(1995) 和 van Seijen(2016) 的研究表明，它们实际上是完全实用的算法。鉴于此，以

及它们在概念上的清晰性和简洁性，我们选择在第二版中突出它们。特别是，我们现在将所有关于逆向观点和资格追溯的讨论推迟到第 12 章。

7.1-2 基于工作对本文进行了随机游动示例中的结果萨顿 (1988)，辛格和萨顿 (1996)。在本章中使用备份图来描述这些和其他算法是新的。

7.3-5 这些部分的开发是基于 Precup, Sutton 的工作，还有 Singh (2000)，Precup, Sutton, Dasgupta(2001)，还有 Sutton, Mahmood, Precup, van Hasselt(2014)。树备份算法是由于 Precup, Sutton 和 Singh(2000)，但它的呈现是新的。

7.6 Q( ) 算法是新到这个文本, 但算法密切相关 De Asis, Hernandez-Garcia, Holland 和 Sutton(2017) 进一步探索。

8. 第八章用表格法进行规划和学习

在本章中，我们提出了一种统一的强化学习方法，它需要一个环境模型，如动态规划和启发式搜索，以及没有模型的方法，如蒙特卡罗和时间差分方法。这些方法分别称为基于模型和无模型强化学习方法。基于模型的方法依赖于计划作为其主要组件，而无模型的方法主要依赖于学习。虽然这两种方法确实存在差异，但也有很大的相似之处。特别地，这两种方法的核心是价值函数的计算。此外，所有的方法都基于对未来事件的展望，计算一个备份值，然后将其作为一个近似值函数的更新目标。在本书的前面，我们提出了蒙特卡罗和时间差方法作为不同的替代方法，然后展示了如何用 n 阶方法来统一它们。本章的目标是对基于模型的和无模型的方法进行类似的集成。在前面的章节中，我们已经确定了它们的不同之处，现在我们来探究它们在多大程度上可以混合在一起。

8.1 模型和规划

通过环境模型，我们指的是任何代理可以用来预测环境对其行为的反应的东西。给定一个状态和一个动作，一个模型生成下一个状态和下一个奖励的预测。如果模型是随机的，那么接下来会有几个可能的状态和下一个奖励，每一个都有发生的概率。有些模型对所有可能性及其概率进行描述; 我们称之为分布模型。其他模型只产生一种可能性，根据概率抽样; 我们称之为样本模型。例如，考虑建模一打骰子的和。一个分布模型将产生所有可能的和及其发生的概率，而一个样本模型将产生一个个体

根据这个概率分布得出的和。MDP 动态规划估计中假定的一种模型，p(s?，r |s a) 是一种分布模型。第 5 章中的 21 点示例使用的模型是一个示例模型。分布模型比样本模型更强，因为它们总是可以用来生产样本。然而，在许多应用程序中，获得样本模型比获得分布模型要容易得多。打骰子就是一个简单的例子。编写一个计算机程序来模拟掷骰子的结果并返回和是很容易的，但是要计算所有可能的和及其概率就会越来越困难，也更容易出错。模型可以用来模拟或模拟体验。给定一个起始状态和动作，一个样本模型将产生一个可能的过渡，一个分布模型将生成所有可能的过渡，并根据它们发生的概率进行加权。给定一个起始状态和策略，示例模型可以生成整个事件，而分布模型可以生成所有可能的事件及其概率。在这两种情况下，我们都说该模型用于模拟环境并产生模拟的经验。规划这个词在不同的领域有不同的用法。我们用这个词来指代任何以模型作为输入并产生或改进与模型环境交互的策略的计算过程: 规划模型政策

在人工智能中，根据我们的定义，有两种不同的规划方法。国家空间规划，包括我们在本书中采用的方法，主要被看作是在国家空间中寻找最优政策或实现目标的最优路径。动作导

致从状态到状态的转换，并且值函数是通过状态来计算的。在我们所说的计划空间规划中，计划是对计划空间的搜索。操作符将一个计划转换为另一个计划，值函数 (如果有的话) 是在计划空间中定义的。规划空间规划包括进化方法和“部分顺序规划”，这是人工智能中常见的一种规划，在规划的所有阶段中，步骤的顺序并没有完全确定。平面空间方法很难有效地应用于随机序列决策问题，而这些问题是强化学习的重点，我们没有进一步考虑它们 (但参见，例如，Russell 和 Norvig, 2010)。我们在本章中提出的统一观点是，所有的国家空间规划方法都有一个共同的结构，这一结构也体现在本书的学习方法中。开发这个视图需要本章的其余部分，但是有两个基本思想:(1) 所有的状态空间规划方法都将计算值函数作为改进策略的关键中间步骤;(2) 通过应用于模拟体验的更新或备份操作来计算值函数。这种共同结构可以如下图所示:

备份模型模拟值经验政策更新备份

动态编程方法显然适合这种结构: 它们使遍历状态空间，为每个状态生成可能的转换的分布。然后使用每个分布来计算备份值 (更新目标) 并更新状态的估计价值。在这一章中，我们认为各种其他的国家空间规划方法也适用于这种结构，不同的方法只在更新的种类、更新的顺序以及保存备份信息的时间上有所不同。以这种方式观看规划方法强调了它们与我们在本书中描述的学习方法的关系。学习和计划方法的核心是通过备份更新操作来估计值函数。不同之处在于，规划使用模型生成的模拟体验，而学习方法使用环境生成的真实体验。当然，这种差异导致了许多其他的差异，例如，如何评估绩效，以及如何灵活地产生经验。但共同的结构意味着许多想法和算法可以在规划和学习之间转换。特别是，在许多情况下，学习算法可以代替计划方法的关键更新步骤。学习方法只需要经验作为输入，在许多情况下，它们既可以用于模拟经验，也可以用于实际经验。下面的框显示了一个简单的规划方法示例，该方法基于一步表 q 学习和样本模型中的随机样本。这种方法, 我们称之为随机样本一步表格 Q-planning, 收敛于最优政策模型在相同条件下, 一步法表格 q 学习收敛于最优政策的实际环境中 (每一对政府行动必须选择在步骤 1 中无限次的, 随着时间的推移和 必须适当减少)。

除了规划和学习方法的统一观点之外，本章的第二个主题是在小的、增量的步骤中进行规划的好处。这使得规划在任何时候都可以被中断或重定向，而很少浪费计算，这似乎是有效地将规划与行为和模型的学习混合在一起的关键需求。如果问题太大而无法精确地解决，即使是在纯粹的规划问题上，以非常小的步骤进行规划也可能是最有效的方法。

8.2 Dyna: 综合规划、表演、学习

当计划在网上完成时，当与环境交互时，会出现许多有趣的问题。从交互中获得的新信息可能会改变模型，从而与计划交互。可能需要以某种方式将规划过程自定义为当前正在审议或预期的国家或决定

在不久的将来。如果决策和模型学习都是计算密集型的过程，那么可用的计算资源可能需要在它们之间进行划分。为了开始研究这些问题，在本节中，我们将介绍 Dyna-Q，一个集成在线规划代理所需的主要功能的简单架构。在 Dyna-Q 中，每个函数都以简单的形式出现。在后面的小节中，我们将详细介绍实现每个函数的一些替代方法以及它们之间的权衡。现在，我们仅仅是为了阐明这些想法并激发你的直觉。在规划代理中，实际体验至少有两个角色: 可以使用它改进模型 (使模型更精确地匹配真实环境)，也可以使用它们直接改进值函数和策略

规划价值/政策

经验模型

模型的学习代理直接 RL 我们在前几章已经讨论过的强化学习方法。前者我们称之为模型学习，后者我们称之为直接强化学习 (直接 RL)。经验、模型、价值和政策之间的可能关系在右边的图表中得到了总结。每个箭头都显示了影响和假定的改进之间的关系。注意经验如何通过模型直接或间接地改进价值功能和策略。后者，有时被称为间接强化学习，涉及到规划。直接法和间接法各有利弊。间接方法通常会更充分地利用有限的经验，从而在较少的环

境交互作用下实现更好的政策。另一方面，直接方法要简单得多，并且不会受到模型设计中的偏差的影响。一些人认为间接方法总是优于直接方法，而另一些人则认为直接方法是人类和动物学习的主要原因。心理学和人工智能的相关辩论涉及认知相对于试错学习的相对重要性，以及商议性规划相对于反应性决策的相对重要性 (从心理学的角度讨论这些问题，见第 14 章)。我们的观点是，在所有这些辩论中，各种选择之间的对比被夸大了，认识到这两方之间的相似之处比反对它们能获得更多的洞察力。例如，在这本书中，我们强调了动态编程和时间性差异方法之间的深层相似性，尽管其中一种方法是为计划而设计的，另一种是为无模型学习而设计的。Dyna-Q 包含了上述图中所示的所有过程，包括计划、行为、模型学习和直接 rl——所有这些过程都在不断地发生。规划方法为第 161 页上的随机样本一步表 q 规划方法。直接 RL 方法是一步表 q 学习。模型学习方法也是基于表格的，并假定环境是确定的。每个转换后圣,→Rt + 1, 圣 + 1, 模型记录了圣表条目, 在 Rt + 1 的预测, 圣 + 1 将确定性跟踪。因此，如果用之前经历过的状态-动作对模型进行查询，则只需返回最后观察到的下一个状态和下一个奖励作为预测。

在规划过程中，Q-planning 算法只从之前经历过的状态-动作对 (步骤 1) 中随机抽取样本，因此模型不会被没有信息的状态-动作对查询。Dyna 代理的总体架构 (其中 Dyna- q 算法就是一个例子) 如图 8.1 所示。中心柱代表了 agent 与环境之间的基本交互，产生了真实体验的轨迹。图左边的箭头表示对真实体验的直接强化学习，以改进价值函数和策略。右边是基于模型的流程。该模型从实际经验中学习，并产生模拟经验。我们使用“搜索控制”这个词来指代由模型生成的模拟体验的启动状态和动作的过程。最后，通过将强化学习方法应用到模拟体验中来实现规划，就像它们真的发生了一样。通常，在 Dyna-Q 中，同样的强化学习方法用于从真实经验中学习，也用于从模拟经验中进行规划。因此，强化学习方法是学习和计划的“最终共同路径”。学习和计划是紧密地结合在一起的，因为它们几乎共享所有相同的机制，只是在经验的来源上有所不同。

图 8.1: 通用 Dyna 体系结构。真正的经验，来回穿梭。环境和策略对策略和价值功能的影响与它们的影响非常相似模拟环境模型产生的经验。

从概念上讲，在 Dyna 代理中，计划、行为、模型学习和直接 RL 同时并行地发生。然而，对于串行计算机上的具体和实现，我们完全指定它们在时间步长内发生的顺序。在 Dyna-Q 中，行为、模型学习和直接 RL 过程只需要很少的计算，我们假设它们只占用一小部分时间。每个步骤的剩余时间可以用于规划过程，这本质上是计算密集型的。让我们假设在执行、模型学习和指导 RL 之后，每一步都有时间来完成

Q-planning 算法的 n 次迭代 (步骤 1-3)。在下面框中 Dyna-Q 的伪代码算法中，模型 (s, a) 表示状态-动作对 (s, a) 的内容 (预测下一个状态和奖励)。直接强化学习，模型学习，和计划分别由步骤 (d)、(e) 和 (f) 来实现。如果省略 (e) 和 (f)，剩下的算法将是一步式的列表 Q-learning。

示例 8.1:Dyna Maze 考虑一下图 8.2 中显示的简单迷宫。在每个 47 个状态中，有 4 个动作，向上、向下、右和左，这些动作对对应的相邻状态有决定性的作用，除非当移动被障碍物或迷宫的边缘阻挡时，在这种情况下，agent 保持在原来的位置。所有转变的奖励都是零，除了那些进入目标状态的，在这个状态下奖励是 +1。在达到目标状态 (G) 之后，代理返回到开始状态，开始新的一集。这是一个折扣, 情景任务与 = 0.95。图 8.2 的主要部分显示了将 Dyna-Q agent 应用于迷宫任务的实验的平均学习曲线。最初的行动值是零, 步长参数 = 0.1, 勘探参数 = 0.1。在动作间的贪婪选择中，关系被随机打破。代理人在计划步骤的数量上是不同的，n，他们按实际步骤执行。对于每个 n，曲线显示了 agent 在每一集中达到目标所采取的步骤的数量，平均超过 30 次的实验重复。在每次重复中，随机数生成器的初始种子在算法之间保持不变。正因为如此，对于所有 n 的值来说，第一个插曲是完全相同的 (大约 1700 步)，它的数据没有显示在图中。在第一集之后，对所有 n 值的性能都有了改进，但是对于更大的值来说则更快了。回想一下，n = 0 代理是一个非规划代理，只使用直接强化学习 (一步表格 q 学习)。这是迄今为止最慢的代理在这个问题上, 尽管参数值 (和) 进行了优化。无计划代理了约 25 集 ( ) 最优性能, 而 n = 5 代理花了五集, 和 n = 50 代理只花了三集。

G 800 年 0 规划步骤
5 计划步骤 50 (direct RL only) 600 年
400 年每一个步骤集
200 年
14
2 20 10 30
集 165 年
行动
40 50

图 8.2: 一个简单的迷宫 (inset) 和 Dyna-Q 的平均学习曲线变化。在他们的计划步骤数量 (n) 每一个真正的步骤。任务是从 S 到 G 越好。

图 8.3 显示了为什么计划代理比非计划代理更快地找到解决方案。如图所示，在第二集中，n = 0 和 n = 50 个代理的策略。在没有计划 (n = 0) 的情况下，每一集都只向策略添加了一个额外的步骤，因此到目前为止，只学习了一个 (最后一个) 步骤。通过计划，在第一集中只学习了一步，但是在第二集中，已经制定了一项广泛的政策，即在这一集结束时，几乎要回到起始状态。此策略是由计划过程构建的，而代理仍然徘徊在开始状态附近。到第三集结束时，将找到一个完整的最优策略，并实现完美的性能。

没有计划 (=0) 和计划 (=50)n。N
G G
年代年代

图 8.3: 计划和非计划的 Dyna-Q 代理在第二集中中途发现的策略。箭头表示每个州的贪婪行为; 如果没有为一个状态显示箭头，那么它的所有动作值都是相等的。黑色方块表示代理的位置。

在 Dyna-Q 中，学习和计划是通过完全相同的算法完成的，在学习的实际经验和计划的模拟经验上操作。因为计划是渐进的，所以把计划和行动结合起来是很简单的。两者都尽可能快地进行。代理总是反应性的，总是深思熟虑的，对最新的感觉信息做出即时反应，但总是在后台进行规划。在后台进行的还有模型学习过程。随着新信息的获取，模型被更新以更好地匹配现实。随着模型的变化，正在进行的规划过程将逐渐计算出与新模型匹配的不同的行为方式。

练习 8.1 非计划方法在图 8.3 中看起来特别糟糕，因为它是一步法; 采用多步引导的方法可以做得更好。你认为第七章中的一种多步骤引导方法也能做得一样好吗强啡肽的方法吗? 解释为什么或为什么不。?

8.3 模型错误时

在前一节中给出的迷宫示例中，模型的变化相对较小。模型一开始是空的，然后只填充正确的信息。总的来说，我们不能指望如此幸运。模型可能是不正确的，因为环境是随机的，并且只有有限的样本被观察到，或者因为模型是用不完美的广义函数逼近来学习的，或者仅仅是因为环境发生了变化，并且它的新行为还没有被观察到。当模型不正确时，规划过程可能会计算一个次优策略。在某些情况下，通过规划计算的次优策略会导致建模误差的发现和修正。当模型乐观地预测比实际可能的更大的回报或更好的状态转变时，就会出现这种情况。计划中的政策试图利用这些机会，并在此过程中发现它们并不存在。示例 8.2: 阻塞迷宫一个迷宫示例演示了这种相对较小的建模错误并从中恢复，如图 8.4 所示。最初，从开始到目标有一条很短的路径，到屏障的右边，如图左上方所示。经过 1000 步后，短路径被“阻塞”，较长的路径沿着屏障的左侧被打开，如图右上角所示。图中显示了 Dyna-Q 代理和增强的 Dyna-Q+ 代理的平均累积报酬。图的第一部分显示了两个 Dyna 代理在 1000 步内找到了短路径。当环境发生变化时，图形变平，表明在这段时间内，这些特工因为在障碍物后面游荡而得不到任

何报酬。然而，过了一段时间，他们找到了新的突破口和新的最优行为。当环境变得比以前更好时，就会出现更大的困难，而以前正确的政策并没有显示出改善。在这些情况下，建模错误可能不会被检测很长时间，如果有的话。

累积奖励 2000 年年代 Dyna-Q

0 1000 年时间步长图 8.4:Dyna 代理在阻塞任务上的平均性能。前 1000 步使用的是左环境，其余的是右环境。Dyna-Q+ 是 Dyna-Q，具有鼓励探索的探索奖金。

累积奖励年代 G G

年代

0 3000 3000

时间步长 400 年

0 Dyna-Q + Dyna-Q

图 8.5:Dyna 试剂的平均性能一个快捷键的任务。左环境用于前 3000 步，剩下的环境。例 8.3: 快捷迷宫图 8.5 所示的迷宫示例说明了由这种环境变化引起的问题。最开始，最优路径是绕过障碍物的左边 (左上)。然而，在 3000 步之后，沿着右边打开一条较短的路径，而不会干扰较长的路径 (右上角)。图表显示，常规的 Dyna-Q 代理从未切换到快捷方式。事实上，它从未意识到它的存在。它的模型说没有捷径可走，所以计划得越多，越不可能走到右边去发现它。即使 -greedy 政策, 很可能一个代理需要很多探索性行为, 发现捷径。

这里的一般问题是探索和开发之间的冲突的另一个版本。在规划环境中，探索意味着尝试改进模型的行动，而开发意味着在当前模型下以最佳方式运行。

我们希望代理能够探索环境中的变化，但不能因此而大大降低性能。就像早期的勘探/开采冲突一样，可能没有既完美又实用的解决方案，但简单的启发式往往是有效的。解决捷径迷宫的 Dyna-Q+ 代理使用了这样的启发式。该代理跟踪每个状态操作对，跟踪自上次在与环境的实际交互中尝试这两个步骤以来的时间间隔。时间过得越久，就越有可能 (我们可能会认为) 这一对的动力学发生了变化，并且模型是错误的。为了鼓励对长期未尝试过的行为进行测试的行为，在涉及这些行为的模拟体验中会给予一个特殊的“奖励”。特别是, 如果过渡的建模奖励是 r, 和过渡尚未在 时间步骤, 然后做计划更新好像转变生产 r + 的奖励 √ 一些小 。这鼓励代理保留测试所有可访问的状态转换，甚至查找长序列动作，以便进行此类测试。当然，所有这些测试都是有代价的，但在许多情况下，就像在快捷迷宫一样，这种计算上的好奇心是值得进行额外探索的。练习 8.2 为什么有探索奖金的 Dyna agent, Dyna- q + 在第一阶段以及阻塞和快捷的第二阶段表现得更好实验吗? ? 练习 8.3 仔细检查图 8.5，发现 Dyna-Q+ 和 Dyna-Q 的差异在第一部分实验中略有缩小。的原因是什么对于这个吗? ? 练习 8.4(编程) 上面描述的探索奖金实际上改变了状态和动作的估计值。这是必要的吗? 使用假设奖金 √ 不更新, 但是只在行动的选择。也就是说, 假设的行动选择的总是问 (圣,)+

(St) 是最大的。执行一个 gridworld 实验测试并演示了这种方法的优缺点替代的方法。? 练习 8.5 第 164 页所示的表格式 Dyna-Q 算法如何修改以处理随机环境? 这个修改如何在不断变化的环境 (如本节中所考虑的) 上执行得很差? 如何修改算法来处理随机环境和变化的环境??

8.4 优先考虑全面

在前几节介绍的 Dyna agent 中，模拟跃迁是在状态-动作对中启动的，这些状态-动作对是从所有先前经历过的对中随机选择的。但统一的选择通常不是最好的; 如果模拟的转换和更新集中在特定的状态-动作对上，那么计划可以更有效。例如, 考虑

Dyna-Q+ 试剂也有另外两种变化。第一，从未有过的行动在上面框中的表格式 Dyna-Q 算法的规划步骤 (f) 中，允许从一个状态进行尝试。第二，这类行为的初始模型是，它们会以零的奖励回到相同的状态。

在第一个迷宫任务的第二集中发生了什么 (图 8.3)。在第二集中开始时，只有直接进入目标的状态-动作对具有正向价值; 所有其他对的值仍然为零。这意味着在几乎所有的转换上执行更新是没有意义的，因为它们将代理从一个零值状态转换为另一个，因此更新没有任何效果。只有在转换到目标之前或之后的状态时进行更新，才会更改任何值。如果模拟的转换是一致生成的，那么在遇到这些有用的转换之前，将进行许多浪费的更新。随着计划的进展，有用的更新区域不断增加，但是如果将重点放在最有用的地方，那么计划的效率仍然远远低于计划。在我们真正的目标——大得多的问题中，状态的数量是如此之多，以至于不集中的搜索将会是极其低效的。这个例子表明，通过从目标状态向后工作，搜索可以有效地集中精力。当然，我们并不是真的想要使用任何特定于“目标状态”的方法。“我们需要为一般的奖励功能而工作的方法。目标状态只是一种特殊情况，便于激发直觉。一般来说，我们不仅希望从目标状态，还希望从任何价值发生变化的状态返回。假设在给定模型的情况下，这些值最初是正确的，就像在发现目标之前在迷宫示例中所做的那样。现在假设代理发现了环境中的变化并更改了一个状态的估计值 (向上或向下)。通常，这意味着许多其他状态的值也应该更改，但是惟一有用的一步更新是那些直接导致值已更改的一个状态的操作。如果这些操作的值被更新，那么前一个状态的值可能会依次变化。如果是这样，那么引入它们的操作需要更新，然后它们的前任状态可能已经改变。通过这种方式，可以从值发生变化的任意状态向后工作，或者执行有用的更新，或者终止传播。这种一般的思想可以称为规划计算的向后聚焦。随着有用更新的前沿向后传播，它通常会迅速增长，产生许多可以有效更新的状态-动作对。但并非所有这些都同样有用。有些州的价值观可能已经发生了很大的变化，而另一些州则几乎没有变化。在随机环境中，估计的过渡概率的变化也会导致变化大小的变化以及需要更新的紧急程度的变化。根据更新的紧迫性对更新进行优先级排序是很自然的，并且按照优先级顺序执行更新。这就是优先级清理背后的思想。每个状态操作对都维护一个队列，如果更新的话，每个状态操作对的估计值都将不会随更改的大小而变化。当队列中的顶部对被更新时，将计算对其每个前任对的影响。如果效果大于某个小阈值，则将这对组合插入到具有新优先级的队列中 (如果队列中有一对之前的条目，那么插入结果只会在队列中保持较高的优先级)。这样，变化的影响被有效地向后传播，直到静止。在下一页的方框中给出了确定性环境的完整算法。

10101010101010101010101010106 107

102 年

04794 186 376 752 1504 3008 6016。

Dyna-Q

优先考虑全面更新到最优解示例 8.4: 在迷宫中按优先级排序的清扫被发现可以极大地提高迷宫任务中找到最优解决方案的速度，通常是 5 到 10 倍。右边是一个典型的例子。这些数据是用于一系列与图 8.2 中所示的结构完全相同的迷宫任务，只不过它们在网格分辨率上有所不同。优先级的扫扫保持了相对于没有优先级的 Dyna-Q 的决定性优势。这两个系统每次环境交互最多更新 n = 5 次。改编自彭和威廉姆斯 (1993)。

将优先级范围扩展到随机环境是很简单的。该模型是通过对每一对国家行动对所经历的次数和下一个州的数量进行统计来维持的。然后，很自然地不使用示例更新来更新每一对，正如我们到目前为止所使用的，而是使用预期的更新，并考虑到所有可能的下一个状态及其发生的可能性。优先级清理只是一种分配计算以提高规划效率的方法，而且可能不是最好的方法。优先级扫描的局限性之一是它使用预期更新，这在随机环境中可能会在低概率转换上浪费大量的计算。正如我们在下一节中所展示的，示例更新

在许多情况下，尽管抽样引入了方差，但计算量较少，能够更接近真值函数。样本更新可以获胜，因为它们将整体的备份计算分解为更小的部分——对应于单个的转换——从而使它更精确地集中在将产生最大影响的部分上。这一想法在 van Seijen 和 Sutton(2013) 引入的“小备份”中达到了逻辑极限。这些是沿着单个转换进行的更新，如示例更新，但是基于没有采样的转换的可能性，如预期的更新。通过选择完成小更新的顺序，可以极大地提高计划效率。我们在本章中建议，所有类型的状态空间规划都可以看作是价值更新的序列，仅在更新的类型、预期或示例、大或小以及更新的顺序上有所不同。在本节中，我们强调了反向聚焦，

但这只是一种策略。例如，另一种办法是根据在现行政策下经常访问的国家能够多容易地联系到这些国家，把重点放在这些国家上，这种政策可以称为前向集中。彭和威廉姆斯 (1993)、巴托 (Barto)、布莱特克 (Bradtke) 和辛格 (Singh)(1995) 研究了前向聚焦的不同版本，接下来几节介绍的方法将其发挥到了极致。

172 年第 8 章: 用表格法进行规划和学习

8.5 预期更新与示例更新

前几节中的示例给出了结合学习和计划方法的可能性范围的一些概念。在本章的其余部分中，我们将从预期更新和示例更新的相对优势开始分析所涉及的一些组件思想。这本书的大部分内容都是关于不同类型的价值功能更新，我们已经考虑了很多种类。目前专注于一步更新，它们主要沿着三个二进制维度变化。前两个维度是更新状态值或动作值，还是估计最优策略的值或任意给定策略的值。这两个维度产生的四类更新近似的四个价值函数, 问 **,q ,v 。的

价值估计预计更新 (DP) 示例更新 (一步 TD)

s

s?

r p 一个

q (年代)

问 ∗(s)v (s)

v∗(年代) s

年代?r 马克斯一个

p 政策评估

值迭代

r s? 年代,

一个? p

q-policy 评价

r s? 年代,

一个? 马克斯 p

核反应能量迭代 s

一个

年代?R

R 年代? 年代,

一个吗?

R 年代? 年代,

马克斯 TD(0)

撒尔沙

q 学习的一个?

图 8.6: 所有步骤的备份图本书考虑的更新。另一个二进制维度是，考虑到可能发生的所有可能的事件，更新是预期的，还是考虑到可能发生的单个示例的示例更新。这三个二进制维度产生了 8 种情况，其中 7 种对应于特定的算法，如图右侧所示。(第八例似乎没有任何有用的更新。) 任何这些一步更新都可以用于规划方法。前面讨论的 Dyna-Q 代理使用 q∗ 样本更新, 但他们也用 q∗ 预计更新, 或预期或样本 q 更新。Dyna-AC 系统一起使用 v 样本更新学习政策结构 (如第 13 章)。对于随机问题，优先级清理总是使用预期的更新之一完成。当我们在第 6 章中介绍一步样例更新时，我们将它们作为预期更新的替代品。在没有分发模型的情况下，不可能进行预期的更新，但是可以使用来自环境的示例转换或示例模型来完成示例更新。在这个观点中隐含的是，如果可能的话，预期的更新比样本更新更可取。但是真的是这样吗? 预期

8.6 预期与样本更新

更新肯定会产生更好的估计，因为它们不会被采样错误破坏，但是它们也需要更多的计算，并且计算常常是规划中的限制资源。为了正确评估预期更新和样本更新的相对优点，我们必须控制它们的不同计算需求。具体性, 考虑预期和样本更新近似问 ∗, 和离散状态和行为的特殊情况, 近似的一览表表示值函数,q, 和一个模型估计的形式动态,p̂(s ?，r |s, a). 状态-动作对 s, a 的预期更新为:

问 (,)←? r s ?, p̂(s ?r | s,)

r + max 一个? Q(s ?, 一个?)

。(8.1)

对应的 s, a 的样本更新，给定一个样本下一个状态和奖励，s ?R(来自模型)，为 q -learning 更新:

Q(,)←Q(s)+

R + max 一个? Q(S ??)−Q(,)

, (8.2)

其中 是平时积极的步长参数。这些预期更新和样本更新之间的差异非常重要，因为环境是随机的，特别是，在给定状态和操作的情况下，许多可能的下一个状态可能会以不同的概率出现。如果只有一个下一个状态是可能的, 然后上面给出的预期和样本更新是相同的 (= 1)。如果有许多可能的下一个状态, 然后可能有显著差异。支持预期更新的是，这是一个精确的计算，导致一个新的 Q(s, a)，其正确性仅受 Q(s) 的正确性所限制。在继承国。样本更新还受到抽样误差的影响。另一方面，示例更新在计算上更便宜，因为它只考虑下一个状态，而不是所有可能的下一个状态。在实践中，更新操作所需的计算通常由计算 Q 的状态-动作对的数量决定。对于一个特定的起始对，s a，设 b 为分支因子。，可能的下一个状态数，s?,p̂(s ?|s, a) > 0)，然后预期的更新将需要大约 b 倍的计算量作为样本更新。如果有足够的时间完成预期的更新，那么由于没有采样错误，结果估计通常比 b 样例更新要好。但是，如果没有足够的时间来完成预期的更新，那么示例更新总是比较可取的，因为它们至少在少于 b 的更新的情况下对价值估计做了一些改进。在许多国家行动对的大问题中，我们往往处于后者。由于有如此多的状态-动作对，所有的期望更新都需要很长时间。在此之前，在许多状态操作对中使用一些示例更新可能比在一些状态操作对中使用预期更新要好得多。给定一个计算工作单元，它是用于一些预期的更新，还是用于 b 倍的示例更新? 图 8.7 显示了一个分析的结果，它给出了这个问题的答案。它将估计误差作为计算时间的函数，用于各种分支因素的预测和样本更新。

计算最大数量一个? Q(s ?, 一个?)

图 8.7: 预期更新和样本更新效率的比较。

b 继承国的可能性相等，初始估计的误差为 1。假设下一个状态的值是正确的，因此预期的更新在完成时将错误减少到零。在这种情况下，样例更新会根据以下内容减少错误

b−1 其中 t 是已经执行的示例更新的数量 (假设样本平均值, 也就是说。 = 1 / t)。关键的观察是，对于中等大小的 b，错误会随着 b 更新的一小部分而急剧下降。对于这些情况，许多状态操作对的值可以显著提高，达到预期更新效果的几个百分点，与此同时，单个状态操作对可以进行预期更新。图 8.7 所示的示例更新的优点可能低估了实际效果。在真正的问题中，继承国的价值将是自己更新的估计数。通过使估计更早地准确，样本更新将具有第二个优势，即继承国所支持的值将更准确。这些结果表明，样本更新很可能优于对具有较大随机分支因子和太多状态无法精确解决的问题的预期更新。以上的分析假设所有 b 可能的下一个状态发生的概率是相等的。相反，假设分布是高度倾斜的，一些 b 状态比大多数更容易发生。比起预期的更新，这是否会加强或削弱示例更新的理由? 支持你的答案。?

8.7 采样轨迹

在本节中，我们将比较分发更新的两种方式。来自动态编程的经典方法是对整个状态 (或状态操作) 空间执行扫描，每次扫描更新每个状态 (或状态操作对)。这是有问题的

在大任务上，因为可能没有时间去完成一次扫描。在许多任务中，绝大多数国家都是无关紧要的，因为它们只在非常差的政策或极低的概率下访问。详尽的扫描隐含地将同样的时间用于状态空间的所有部分，而不是集中在需要的地方。正如我们在第 4 章中所讨论的，详尽的扫掠和它们所暗示的所有状态的平等处理并不是动态规划的必要属性。原则上，更新可以以任何你喜欢的方式发布 (为了保证聚合，所有状态或状态-动作对必须以无限次的限制访问; 虽然在下面的第 8.7 节中讨论了这个异常，但是实际上经常使用彻底的扫描。第二种方法是根据一定的分布从状态或状态作用空间中抽取样本。就像 Dyna-Q 试剂一样，我们可以均匀地取样，但这也会遇到一些与彻底扫瞄相同的问题。更吸引人的是根据策略分发分发分发更新，即根据跟踪当前策略时观察到的分发。这种分布的一个优点是很容易生成; 您只需按照当前策略与模型进行交互。在情景性任务中，一个人从开始状态 (或根据开始状态分布) 开始并模拟到结束状态。在一个持续的任务中，你可以从任何地方开始，并不断地模拟。无论哪种情况，模型都会给出示例状态转换和奖励，而当前策略给出示例操作。换句话说，一个人模拟显式的个人轨迹，并在沿途遇到的状态或状态对上执行更新。我们称之为产生经验和更新轨迹抽样的方式。很难想象有什么有效的方式来分配更新，根据政策的分配，除了通过轨道抽样。如果有一个显式的策略分布表示，那么就可以遍历所有状态，根据策略分布对每个状态的更新进行加权，但是这又给我们留下了所有彻底清除的计算成本。也许可以从分布中取样和更新单个状态-动作对，但即使这可以有效地完成，这对模拟轨迹有什么好处呢? 甚至不太可能知道显式的政策分配。每当策略发生变化时，分布就会发生变化，计算分布需要与完整的策略评估相当的计算。考虑到其他的可能性，使得轨迹采样看起来既高效又优雅。更新的策略分发是一个好的吗? 直觉上，这似乎是个不错的选择，至少比均匀分布要好。例如，如果你正在学习下象棋，你要学习的是在真正的游戏中可能出现的位置，而不是棋子的随机位置。后者可能是有效的状态，但能够准确地评估它们是一种不同的技能，从评估真实游戏中的位置。我们还将在第二部分中看到，当使用函数逼近时，策略上的分布具有显著的优势。无论是否使用函数逼近，我们都可以期望专注于策略以显著提高计划的速度。专注于策略分配可能是有益的，因为它会导致大量、无趣的空间部分被忽略，或者可能是有害的，因为它会导致空间中相同的旧部分不断更新。我们进行了一个小

通过实验对效果进行实证评估。为了隔离更新分发的影响，我们使用了完全一步的预期表格更新，如 (8.1) 所定义。在统一的情况下, 我们骑车穿过所有的政府行动对, 更新每个到位, 我们模拟集在政策的情况下, 所有从相同的状态, 更新每一对政府行动发生在当前 -greedy 政策 (= 0.1)。这些任务是不间断的任务，随机生成如下。从 |S| 的每个状态中，都可能有两个动作，每个动作都会导致 b 的下一个状态，所有的可能性都是相等的，每个状态-动作对的 b 状态的随机选择都不一样。分支因子 b，对于所有的状态-行为对都是一样的。此外，在所有的转变中，有 0.1 个转变到最终状态的概率，结束了这个插曲。每个过渡的期望回报是从均值 0 和方差 1 的高斯分布中选择的。

b = 10 b = 3 b = 1
b = 1 我 on-pol cy
我 on-pol cy 统一的
统一的 0 1 2 3
开始状态的值贪婪的政策
0 5000 10000 15000 20000
计算时间，完全备份。
0 1 2 3
开始状态的值贪婪的政策
50 万，10 万，15 万，20 万

计算时间，完全备份统一的
统一的在政策
在政策
预计更新
预计更新
10000 个国家

图 8.8: 在状态空间上均匀分布的相对效率，而不是集中在模拟的 on-policy 轨迹上。从相同的状态开始。结果用于随机生成的两个大小和不同分支的任务因素,b。在规划过程可以在任何时候停止并详尽计算 v˜(s0), 一开始状态的真正价值在贪婪的政策下,˜, 鉴于当前行为价值函数 Q, 视为代理会做如何的新一集它是贪婪地 (同时假设模型是正确的)。右图的上部显示了 200 多个样本任务的平均结果，这些任务具有 1000 个状态和 1、3 和 10 个分支因子。发现的策略的质量被绘制成完成的预期更新数量的函数。在所有情况下，根据政策分布进行抽样，会使最初的规划速度更快，而长期的规划速度较慢。在较小的分支因子下，效果更强，更快规划的初始阶段更长。在其他实验中，我们发现随着状态数的增加，这些效应也变得更强。例如，图的下半部分显示了具有 10,000 个状态的任务的分支因子 1 的结果。在这种情况下，专注于政策的优势是巨大而持久的。所有这些结果都是有道理的。在短期内，根据政策分布进行抽样有助于集中注意与政策分布相近的州

一开始的状态。如果有许多状态和一个小的分支因子，这种影响将是巨大和持久的。从长远来看，关注政策上的分布可能会造成伤害，因为通常出现的状态都已经有了正确的值。对它们进行采样是无用的，而对其他状态进行采样实际上可能执行一些有用的工作。这大概就是为什么详尽、不集中的方法从长期来看效果更好，至少在小问题上是这样。这些结果并不是决定性的, 因为他们只在一个特定的生成问题, 随机的方式, 但他们认为, 根据政策分布抽样可以为大问题是一个很大的优势, 尤其是对问题的一小部分在政策下的政府行动空间访问分布。练习 8.7 图 8.8 中的一些图形在它们的早期部分似乎是扇形的，特别是 b = 1 的上图和均匀分布。你认为这是为什么? 数据的哪些方面支持你的假设? 吗? 练习 8.8(编程) 复制实验结果如图 8.8 中所示，然后尝试相同的实验，但 b = 3。讨论了你的结果的意义。?

8.8 实时动态规划

实时动态规划 (RTDP) 是动态规划的价值迭代算法 (DP) 的一种基于策略的轨迹采样算法。因为它与传统的基于缓存的策略迭代密切相关，RTDP 以一种特别清晰的方式说明了在策略轨迹抽样中所能提供的一些优势。RTDP 通过按 (4.10) 定义的期望表格值迭代更新来更新实际或模拟轨迹中访问的状态的值。它基本上是生成图 8.8 所示的策略结果的算法。RTDP 与传统 DP 的密切联系使得利用已有的理论可以得到一些理论结果。RTDP 是第 4.5 节中描述的异步 DP 算法的一个示例。异步 DP 算法不是按照对状态集的系统扫描来组织的; 它们以任何顺序更新状态值，使用其他状态的值。在 RTDP 中，更新顺序由访问真实轨迹或模拟轨迹的顺序状态决定。开始状态无关紧要的国家: 在任何最优策略下，从任何起始状态都无法到达

相关的国家在一些最优策略下从某个起始状态可达如果轨迹可以只从一组指定的开始, 如果你有兴趣在预测问题对于一个给定的政策, 然后在政策轨迹采样算法可以完全跳过, 不能从任何给定的政策达成的开始状态: 这些国家预测问题是无关紧要的。对于控制问题，如果目标是找到最优策略而不是评估给定的策略，那么很可能存在不可能的状态

由任何起始状态的最优策略达成，不需要为这些不相关的状态指定最优操作。我们需要的是一个最优部分策略，即对相关国家最优的策略，但可以为不相关国家指定任意操作，甚至是未定义的策略。但是要找到这样一个具有 on-policy 轨迹抽样控制方法的最优部分策略 (如 Sarsa(第 6.4 节))，通常需要访问所有的状态-动作- - -即使是那些最终会被证明是无关的- - - -的组合，次数是无限的。这可以通过使用 explore begin(第 5.3 节) 来实现。对于 RTDP 也是如此: 对于开始探索的情景任务，RTDP 是一种异步的值迭代算法，它收敛于具有折扣的有

限 MDPs(以及在特定条件下的未折扣情况) 的最优策略。与预测问题的情况不同，通常不可能停止更新任何状态或状态-动作对，如果收敛到最优策略是重要的。RTDP 最有趣的结果是，对于满足合理条件的某些类型的问题，RTDP 保证在不无限频繁地访问每个国家，甚至根本不访问某些国家的情况下，找到对相关国家最优的政策。事实上，在一些问题中，只有一小部分国家需要访问。对于状态集很大的问题，这是一个很大的优势，在这种情况下，即使是一次扫描也不可行。这一结果所包含的任务是多边开发银行的不间断任务，目标状态吸引人，产生零回报，如第 3.4 节所述。在真实轨迹或模拟轨迹的每一步，RTDP 都会选择一个贪婪的动作 (随机断线)，并将预期值迭代更新操作应用到当前状态。它还可以在每个步骤中更新任意集合的其他状态的值; 例如，它可以更新从当前状态开始的有限视界前视搜索中访问的状态的值。对于这些问题，每一集从开始状态的集合中随机抽取，并以目标状态结束，RTDP 与概率为 1 的策略，对于所有相关的状态都是最优的:1) 每个目标状态的初始值是 0,2) 存在至少一个政策, 保证将达到一个目标状态的概率从任何一个开始状态,3) 所有奖励过渡件州严格负, 和 4) 所有的初始值都等于或大于其最优值 (可满足通过简单设置所有状态的初始值为零)。具有这些性质的任务是随机最优路径问题的例子，这些问题通常用成本最小化来表示，而不是像我们这里所做的那样，用报酬最大化来表示。在我们的版本中，最大限度地增加负回报等于最小化从起始状态到目标状态的路径成本。这种任务是最短时间控制任务的例子, 在每个时间步需要达成的目标产生 -1 的奖励, 或高尔夫 3.5 节中的示例问题, 其目标是用最少的打洞中风。

例 8.6: 跑道上的 RTDP 练习的跑道问题 5.12(第 111 页) 是一个随机最优路径问题。将 RTDP 和传统的 DP 值迭代算法对一个跑道问题进行了比较，说明了基于策略的轨迹抽样的一些优点。从练习中回想一下，一个代理人必须学会如何像图 5.5 中所示的那样在转弯时驾驶一辆车，并在赛道上尽快越过终点线。起始状态是起始线上所有的零速度状态; 目标状态是所有可以在一个时间内到达的状态，通过在赛道内穿越终点线。与练习 5.12 不同，这里没有对汽车速度的限制，因此状态集可能是无限的。然而，可以通过任何策略从起始状态集获得的状态集是有限的，可以被认为是问题的状态集。每一集都以随机选择的开始状态开始，当赛车越过终点线时结束。奖励是 -1 每一步直到车穿过终点线。如果汽车撞到赛道边界，它会被移回一个随机的开始状态，然后情节继续。类似于图 5.5 左边的小型赛马场的赛马场，任何策略都可以从起始状态到达 9,115 个州，其中只有 599 个州是相关的，这意味着它们可以通过一些最优策略从起始状态到达。(有关国家的数目是通过计算访问的国家，同时执行 107 集的最佳行动来估计的。) 下表比较了用常规 DP 和 RTDP 解决这个问题。这些结果是超过 25 次的平均值，每一次都以不同的随机数开始。传统 DP 在本例中是值迭代使用状态集的详尽的清洁工, 值更新的一个州, 这意味着每个状态的更新使用其他州的最新值 (这是高斯-赛德尔迭代版本的价值, 这是发现大约两倍雅可比版本在这个问题上。参见 4.8 节)。没有特别注意更新的顺序; 其他排序可能会产生更快的收敛速度。两个方法每次运行的初始值都为 0。DP 是判断聚合时最大变化状态值扫描不到 10−4, 和 RTDP 被判断聚合时的平均时间跨越终点线超过 20 集似乎在一个渐近稳定的步骤。这个版本的 RTDP 只更新每个步骤上当前状态的值。

DP RTDP 平均计算收敛的平均数量更新收敛平均数量的更新每集

- - - - - - 4000 集 127600 31.9 98.45 80.51 3.18 - - - - - -

这两种方法生成的策略平均需要 14 到 15 步才能跨越终点线，但 RTDP 只需要 DP 所做的更新的一半。这是 RTDP 政策轨迹抽样的结果。而每个状态的值都被更新

在 DP 的每一次扫描中，RTDP 都关注较少的状态更新。在平均运行中，RTDP 更新了 98.45RTDP 的另一个优点是, 随着价值函数趋于最优值函数 v_*, 代理生成轨迹方法所使用的政策最优政策, 因为它总是贪婪的当前值函数。这与传统价值迭代的情况相反。在实践中，当值函数在一次扫描中仅发生很小的变化时，值迭代就终止了，这就是我们终止值迭代以获得上面表格中的结果的方式。在这一点上, 价值函数密切接近 v_*, 贪婪的政策接近最优的政策。但是，在价值迭代终止之前，对最新值函数贪婪的策略可能是最优的，或者几乎是最优的。(从第四章回忆, 最优策略可以贪婪对许多不同的价值函数, 不仅 v_*)。在值迭代收敛之前检查最优策略的出现并不是传统的 DP 算法的一部分，需要大量的额外计算。在赛马场的例子中,

通过运行许多测试集每次 DP 扫描后, 用行动选择贪婪地根据扫描的结果, 可以估计最早的点的 DP 计算近似最优评价函数是足够好, 这样相应的贪婪的政策几乎是最优的。对于这个 racetrack，在 15 次值迭代或者 136,725 次值迭代更新之后，会出现一个接近最优策略。这是大大低于 252784 DP 的更新需要收敛于 v∗, 但窗台上超过 127600 年的更新 RTDP 必需的。虽然这些仿真当然不是 RTDP 与传统的基于扫描的值迭代的最终比较，但它们说明了基于策略的轨迹抽样的一些优点。常规的值迭代持续更新所有状态的值，而 RTDP 则强烈关注与问题目标相关的状态子集。随着学习的继续，这种关注变得越来越狭隘。因为 RTDP 的收敛定理适用于模拟，我们知道 RTDP 最终只关注相关的状态，即。，在制定最佳路径的州。RTDP 几乎实现了最优控制，其计算量约为基于扫描的值迭代所需计算量的 50

8.9 在决策时进行规划

计划至少可以用两种方式。我们在这一章中所考虑的，以动态规划和 Dyna 为代表的一种方法是，利用规划，在从模型 (样本或分布模型) 获得的模拟经验的基础上，逐步改进策略或价值函数。然后，选择动作就是比较当前状态从表中获得的动作值，我们已经考虑过这种情况，或者用下面第二部分中考虑的近似方法计算数学表达式。在为任何当前的状态 St 选择一个动作之前，计划在改进表条目或

数学表达式，需要为许多状态选择动作，包括 st 使用这种方式，规划不关注当前状态。我们称这种方式的规划为背景规划。另一种使用规划的方法是在遇到每个新的状态 St 后开始并完成它，作为一个计算，其输出是在; 在下一个步骤中，规划从 St+1 重新开始，生成 At+1，以此类推。最简单, 几乎退化, 这样的例子使用计划是当状态值, 并选择一个行动通过比较模型预测未来状态的值为每个行动 (或通过比较的值 afterstates 如井字的例子在第 1 章), 更普遍的是, 规划以这种方式使用可以看远比领先一步, 评估行动的选择导致不同的预测状态和轨迹的奖励。与规划的第一次使用不同，这里的规划侧重于特定的状态。我们称之为决策时间规划。这两种思考方式借助模拟经验, 逐步完善政策或价值函数, 或者使用模拟的经验对当前国家可以选择一个行动自然和有趣的方式混合在一起, 但是他们往往是单独研究, 这是一个好方法首先了解他们。现在让我们来仔细看看决策时间规划。即使计划只在决策时完成，我们仍然可以将其视为从模拟体验到更新和值，最终到策略。只是现在的值和策略是特定于当前状态和可用的操作选择，以至于规划过程创建的值和策略通常在用于选择当前操作之后被丢弃。在许多应用程序中，这并不是很大的损失，因为有很多州，而且我们不太可能在很长一段时间内回到同一个州。一般来说，人们可能想要同时做两件事: 把计划的重点放在当前的状态上，并将计划的结果存储起来，以便在以后返回到相同的状态时能够走得更远。决策时间规划在不需要快速响应的应用程序中最有用。例如，在下棋程序中，一个棋子可能被允许进行几秒或几分钟的计算，而强程序可能计划在这段时间内进行几十次棋步。另一方面，如果低延迟动作选择是优先级，那么通常情况下，最好在后台进行规划，以计算可以快速应用于每个新遇到的状态的策略。

8.10 启发式搜索

人工智能中经典的状态-空间规划方法是决策-时间规划方法，统称为启发式搜索。在启发式搜索中，对于遇到的每个状态，都考虑一个可能延续的大树。将近似值函数应用于叶节点，然后备份到根的当前状态。搜索树中的备份是在预期的更新一样的高峰 (v∗ 和 q∗) 在本书中讨论。备份停止在当前状态的状态操作节点处。一旦计算出这些节点的备份值，最好选择它们作为当前操作，然后全部。备份值被丢弃。在传统的启发式搜索中，不需要通过改变近似值函数来保存备份值。事实上，价值函数一般是由人设计的，不会因为搜索而改变。然而，很自然地，我们会考虑允许值函数随时间而改进，使用启发式搜索中计算的备份值或本书中介绍的任何其他方法。从某种意义上说，我们一直采用这种方法。我们的贪婪、 -greedy UCB(2.7

节) 行为选择方法并不是与启发式搜索不同, 尽管规模较小。例如，要计算给定一个模型和一个状态值函数的贪心行为，我们必须从每个可能的行为到每个可能的下一个状态，考虑到奖励和估计值，然后选择最佳的行为。就像传统的启发式搜索一样，这个过程计算可能操作的备份值，但不尝试保存它们。因此，启发式搜索可以被看作是贪婪策略思想的延伸，超越了单个步骤。比一步更深入的搜索是为了获得更好的操作选择。如果一个人有一个完美的模型和一个不完美的行为价值函数，那么实际上深入的搜索通常会产生更好的策略。2 . 当然，如果搜索一直持续到事件结束，那么不完全值函数的影响就被消除了，以这种方式确定的行为必须是最优的。如果足够的搜索深度 k k 很小, 然后将相应的行动接近最优。另一方面，搜索越深入，需要的计算就越多，通常会导致响应时间变慢。Tesauro 的特级西洋双陆棋玩家 TD-Gammon(第 16.1 节) 提供了一个很好的例子。本系统利用 TD 学习通过多款自玩游戏学习后态值函数，采用启发式搜索的形式进行运动。作为一个模型，TD-Gammon 使用事先知道掷骰子的概率，并假设对手总是选择 TD-Gammon 认为最适合的动作。Tesauro 发现，启发式搜索越深入，TD-Gammon 的移动越好，但每次移动都需要更长的时间。Backgammon 有一个很大的分支因素，但必须在几秒钟内移动。只有在前面有选择地搜索几个步骤是可行的，但即使这样，搜索结果也显著地改善了操作选择。我们不应该忽视启发式搜索关注更新的最明显的方式: 当前状态。启发式搜索之所以有效，很大程度上是因为它的搜索树密切关注可能立即跟随当前状态的状态和操作。也许你一生中下棋的时间比西洋跳棋要多，但是当你下西洋跳棋时，考虑你的特定的西洋跳棋位置、你可能的下一个棋以及下一个棋子的位置是值得的。无论您如何选择操作，这些状态和操作是更新的最高优先级，并且您最迫切地希望您的近似值函数是准确的。您的计算不仅应该优先用于即将发生的事件，还应该优先用于有限的内存资源。例如，在国际象棋中，有太多可能的位置来存储每个位置的不同价值估计，但是基于启发式搜索的国际象棋程序可以很容易地存储它们遇到的数百万个位置的不同估计

有一些有趣的例外 (见，例如，Pearl, 1984)。

从一个单一的位置向前看。这种对内存和计算资源的极大关注可能是启发式搜索如此有效的原因。更新的发布可以以类似的方式进行更改，以关注当前状态及其可能的继任者。作为一个极限情况，我们可能会使用启发式搜索的方法来构建一个搜索树，然后执行自底向上的单步更新，如图 8.9 所示。如果以这种方式对更新进行排序并使用表格表示，那么将实现与深度优先启发式搜索完全相同的整体更新。任何状态空间搜索都可以被看作是将大量单个单步更新拼凑在一起。因此，在深度搜索中观察到的性能改进并不是因为使用了多步更新。相反，这是由于更新的焦点和集中在当前状态下的状态和操作上。通过投入大量与候选动作相关的计算，决策时间规划可以产生比依赖于不集中的更新更好的决策。

图 8.9: 启发式搜索可以实现为一个单步更新序列 (在这里用蓝色表示)，从叶子节点备份到根节点的值。这里显示的顺序是用于选择深度优先搜索。

8.11 Rollout 算法

Rollout 算法是一种基于蒙特卡罗控制的决策时间规划算法，用于模拟轨迹，这些轨迹都是从当前环境状态开始的。他们通过对许多模拟轨迹的平均来估计给定政策的行动值，这些轨迹从每个可能的行动开始，然后遵循给定的政策。当动作价值估计值被认为足够准确时，动作 (或其中一个动作)

具有最高估计值的操作将被执行，然后流程将从产生的下一个状态中重新执行。正如 Tesauro 和 Galperin(1997) 所解释的，“rollout”一词来自于估计一个 backgammon 位置的值，即“rollout”。“滚出”指的是在游戏结束前，玩家通过随机生成的掷骰子序列多次移动的位置。与第五章中描述的蒙特卡罗控制算法,rollout 算法的目标是估计一个完整的优化行为价值函数, 问 $*$, 或一个完整的行为价值函数, q 对于给定的政策。相反，它们只对每个当前状态和一个通常称为 rollout 策略的给定策略生成行为值的蒙特卡罗估计。作为决策时间规划算法，推出算法立即利用这些行动价值估计，然后丢弃它们。这使得 rollout 算法实现起来相

对简单，因为不需要为每个状态-动作对抽样结果，也不需要在状态空间或状态-动作空间中估计一个函数。那么，推出算法实现了什么呢?4.2 节中描述的政策改进定理告诉我们, 给定的任意两个政策 ？是相同的除了 ?(s)= a ？ = (s) 对某些国家, 如果 q (,) v (s), 那么政策 ？是一样好, 或者更好, 比 。此外, 如果不严格, 那么 吗? 实际上是比 。这适用于 rollout 算法是当前状态和 推出政策。模拟轨迹产生估计的平均回报 q (年代,?) 为每个行动? (s)。然后选择一个行动的政策年代最大化这些估计, 此后 是一个很好的候选人的政策改善了 。结果类似于第 4.3 节中讨论的动态编程策略迭代算法的一个步骤 (尽管它更像是第 4.5 节中描述的异步值迭代的一个步骤，因为它只改变当前状态的操作)。换句话说，推出算法的目的是改进推出策略; 不是为了找到最优策略。经验表明，推出算法可以非常有效。例如，Tesauro 和 Galperin(1997) 对 rollout 方法所产生的西洋双陆棋游戏能力的显著提高感到惊讶。在某些应用程序中，即使 rollout 策略完全是随机的，也可以使用 rollout 算法获得良好的性能。但是改进策略的性能取决于推出策略的属性和蒙特卡罗值估计产生的操作的排序。直觉告诉我们，推出策略越好，价值估计越准确，由推出算法生成的策略就可能越好 (但是参见 Gelly 和 Silver, 2007)。这涉及到重要的权衡，因为更好的推出策略通常意味着需要更多的时间来模拟足够多的轨迹，以获得良好的价值评估。作为决策时间规划方法，rollout 算法通常必须满足严格的时间约束。rollout 算法所需的计算时间取决于数量的行动必须被评估为每一个决定, 时间步的数量所需要的模拟轨迹获得有用的示例返回, 所花费的时间推出政策决策和模拟轨迹的数量需要获得良好的蒙特卡罗行为价值的估计。

在任何 rollout 方法的应用中，平衡这些因素都很重要，不过有几种方法可以减轻这种挑战。因为蒙特卡罗试验是相互独立的，所以可以在单独的处理器上并行运行许多试验。另一种方法是截断短于完整集的模拟轨迹，通过存储的评估函数纠正截断的返回 (这使我们在前面几章中关于截断的返回和更新的所有内容发挥作用)。也有可能, 因为 Tesauro 和加尔佩林 (1997) 表明, 监控蒙特卡罗模拟和修剪掉的候选人的行动不太可能是最好的, 或其值接近, 当前最好的选择他们将没有真正的区别 (尽管 Tesauro 和加尔佩林指出, 这将使一个并行实现)。我们通常不认为推出算法是学习算法，因为它们不维护对价值或策略的长期记忆。然而，这些算法利用了我们在这本书中强调的强化学习的一些特点。作为蒙特卡罗控制的实例，它们通过对一组样本轨迹的返回进行平均来估计动作值，在这种情况下，模拟与环境样本模型的相互作用轨迹。这样，它们就像增强学习算法一样，避免了通过轨迹抽样对动态规划的彻底扫荡，避免了依赖于样本 (而非预期) 更新的分布模型的需要。最后，推出算法利用策略改进特性，对估计的操作值执行贪婪操作。

8.12 特卡罗树搜索

蒙特卡罗树搜索 (MCTS) 是最近一个非常成功的决策时间规划的例子。在它的基础上，MCTS 是如上所述的一种推出算法，但是增加了一种方法来积累蒙特卡罗模拟得到的值估计，以便连续地将模拟引向更有价值的轨迹。MCTS 在很大程度上促进了计算机的发展，从 2005 年的业余水平下降到 2015 年的大师水平 (6 dan 或更多)。基本算法的许多变体已经被开发出来，包括我们在第 16.6 节中讨论的一个变体，它对于程序 AlphaGo 在 2016 年击败一名 18 届世界围棋冠军至关重要。未经中华人民共和国交通部已经证明是有效的在各种各样的竞争环境中, 包括通用游戏 (例如, 看到 Finnsson Björnsson,2008;Genesereth 和 Thielscher, 2014)，但不限于游戏; 如果环境模型足够简单，可以进行快速的多步仿真，那么它对单代理的连续决策问题是有效的。MCTS 在遇到每个新状态后执行，以选择该状态的代理动作; 它再次执行，为下一个状态选择操作，以此类推。与 rollout 算法一样，每次执行都是一个迭代过程，它模拟许多轨迹，从当前状态开始，运行到终端状态 (或者直到折扣使任何进一步的奖励作为回报的贡献变得微不足道)。MCTS 的核心思想是连续地将多个模拟从当前状态开始

扩展从早期模拟得到高评价的轨迹的初始部分。MCTS 不必保留从一个操作选择到下一个操作选择的近似值函数或策略，但在许多实现中，它保留了可能对下一次执行有用的选定

操作值。在大多数情况下，模拟轨迹中的操作是使用一个简单的策略生成的，通常称为 rollout 策略，因为它用于更简单的 rollout 算法。当 rollout 策略和模型都不需要大量计算时，可以在很短的时间内生成许多模拟轨迹。与任何表列蒙特卡罗方法一样，状态-动作对的值被估计为这对 (模拟) 返回值的平均值。蒙特卡罗值估计只针对最可能在几个步骤中达到的状态-动作对的子集，它形成了根在当前状态的树，如图 8.10 所示。基于模拟轨迹的结果，MCTS 通过添加表示状态的节点来逐步扩展树。任何模拟的轨迹都会经过这棵树，然后在某个叶子节点上退出。在树和叶子节点上，rollout 策略用于操作选择，但是在树内的状态中，更好的情况是可能的。对于这些州，我们至少有一些行动的价值估计，因此我们可以在他们之间使用一种被称为“树政策”的明智政策来平衡探索。

选择模拟扩张备份重复而时间仍

图 8.10: 蒙特卡罗树搜索。当环境改变到一个新的状态时，MCTS 在需要增量地选择操作之前，执行尽可能多的迭代构建一个根节点表示当前状态的树。每个迭代由四个元素组成操作选择，扩展 (虽然可能在某些迭代中跳过)，模拟，和备份，就像文中解释的那样，用树中的粗体箭头来说明。改编来自 Chaslot, Bakkes, Szita 和 Spronck(2008)。

与剥削。例如, 树策略可以选择操作使用 -greedy 或联合选择规则 (第二章)。更详细地说，MCTS 的基本版本的每个迭代包括以下四个步骤，如图 8.10 所示:

1。选择。从根节点开始，基于附加到树边缘的操作值的树策略将遍历树以选择叶子节点。

2。扩张。在一些迭代中 (取决于应用程序的细节)，通过添加一个或多个通过未探索的操作从选定节点到达的子节点，树从选定的叶子节点扩展。

3 所示。模拟。从所选的节点，或者从其新增的子节点 (如果有的话) 中，将运行一个完整的事件的模拟，并使用 rollout 策略所选择的操作。结果是一个蒙特卡罗试验，首先由树策略选择动作，然后再由树推出策略选择树之外的动作。

4 所示。备份。模拟事件生成的返回被备份，以更新或初始化在此 MCTS 迭代中由树策略遍历的树的边缘附加的操作值。不为树之外的 rollout 策略访问的状态和操作保存任何值。图 8.10 显示了从模拟轨迹的终端状态直接到开始执行 rollout 策略的树中的状态操作节点的备份 (尽管通常情况下，整个模拟轨迹的返回都备份到这个状态操作节点)。

MCTS 继续执行这四个步骤，每次从树的根节点开始，直到没有剩余的时间，或者其他计算资源耗尽。然后，最后，根据依赖于树中累积的统计信息的某种机制选择根节点 (仍然表示环境的当前状态) 的操作; 例如，它可能是一个操作，其操作值可能是根状态中所有可用操作中最大的，或者可能是访问计数最大的操作，以避免选择离群值。这是 MCTS 实际选择的动作。在环境转换到新状态之后，MCTS 将再次运行，有时从表示新状态的单个根节点的树开始，但通常从包含该节点的任何后代的树开始，这些后代来自于以前执行 MCTS 时所构建的树; 所有剩余的节点以及与它们相关的操作值都将被丢弃。MCTS 最初被提议在播放双人竞技游戏 (如围棋) 的程序中选择动作。在游戏中，每个模拟的场景都是游戏的一个完整的游戏，在游戏中，两个玩家都选择树的动作和推出的策略。第 16.6 节描述了 AlphaGo 程序中使用的 MCTS 的一个扩展，该扩展将 MCTS 的蒙特卡罗评估与深度人工神经网络通过自玩强化学习获得的动作值结合起来。将 MCTS 与我们在本书中描述的强化学习原理相结合，可以让我们了解它是如何获得如此令人印象深刻的结果的。MCTS 是一种基于蒙特卡罗控制的决策时间规划算法

从根状态开始; 也就是说，它是前一节描述的一种 rollout 算法。因此，它受益于在线的、增量的、基于样本的价值评估和策略改进。除此之外，它还保存附加到树边缘的动作值估计，并使用强化学习的示例更新对它们进行更新。它的作用是将蒙特卡罗试验集中在轨迹上，其初始段与先前模拟的高回报轨迹相同。此外，通过逐步扩展树，MCTS 有效地扩展查找表来存储部分动作值函数，并将内存分配给在高收益样本轨迹的初始段中访问的状态操作对的估计值。因此，MCTS 避免了全局近似行为价值函数的问题，同时保留了利用过去经验指导探索的好处。MCTS 的决策时间规划取得了惊人的成功，对人工智能产生了深远的影响，许多研究人员正在研究对基本程序的修改和扩展，以便在游戏和单智能体应用中使用。

8.13 本章小结

规划需要一个环境模型。分布模型由下一个状态的概率和可能的行为的奖励组成; 一个样本模型产生单个转换和根据这些概率产生的奖励。动态编程需要一个分布模型，因为它使用预期更新，这涉及到对所有可能的下一个状态和奖励的计算期望。另一方面，示例模型是模拟与环境交互所需要的，在此过程中，可以使用示例更新，如许多增强学习算法所使用的更新。样本模型通常比分布模型更容易获得。我们提出了一种观点，强调规划最优行为和学习最优行为之间惊人的密切关系。两者都涉及估计相同的值函数，在这两种情况下，都很自然地在一系列小型备份操作中逐步更新估计。这使得集成学习和规划过程变得非常简单，只需允许两者更新相同的估计值函数。此外，任何学习方法都可以通过将它们应用于模拟 (模型生成的) 体验而不是实际体验而转化为规划方法。在这种情况下，学习和计划变得更加相似; 它们可能是相同的算法，在两个不同的经验来源上运行。将增量规划方法与行为和模型学习相结合是很简单的。计划、表演和模型学习以循环的方式进行交互 (如第 162 页的图表)，每一个都产生了其他需要改进的东西; 它们之间没有其他的相互作用是必需的或禁止的。最自然的方法是让所有进程异步并行地进行。如果进程必须共享计算资源，那么可以几乎任意地处理分割——由任何对当前任务最方便和有效的组织来处理。在本章中，我们讨论了国家空间规划方法之间的一些变化维度。一个维度是更新大小的变化。的

更新越小，计划方法就可以越增量。在最小的更新中有一步样本更新，就像在 Dyna 中一样。另一个重要的方面是更新的分布，也就是搜索的重点。优先级的全面扫荡集中在最近改变其价值观的国家的前身。策略上的轨迹抽样集中在代理控制环境时可能遇到的状态或状态对上。这允许计算跳过与预测或控制问题无关的部分状态空间。实时动态编程是一种基于策略的价值迭代采样版本，它展示了该策略相对于传统的基于扫描的策略迭代的一些优势。规划还可以从相关的状态开始，例如在代理-环境交互过程中实际遇到的状态。最重要的形式是在决策时进行规划，即作为行动选择过程的一部分。在人工智能中研究的经典启发式搜索就是一个例子。其他的例子还有推出算法和蒙特卡罗树搜索，它们得益于在线的、增量的、基于样本的价值估计和策略改进。

8.14 第一部分概述: 尺寸

这一章结束了这本书的第一部分。在这篇文章中，我们试图把强化学习描述成一套贯穿方法的连贯的思想。每个想法都可以看作是方法变化的一个维度。这些维度的集合跨越了大量可能的方法。通过在维度层面上探索这个空间，我们希望获得最广泛、最持久的理解。在这一节中，我们使用方法空间中的维数概念来概括本书迄今为止发展起来的强化学习的观点。到目前为止，我们在本书中探索的所有方法都有三个共同点: 第一，它们都寻求估算价值函数; 第二，它们都通过沿着实际或可能的状态轨迹备份值来操作; 第三，它们都遵循广义策略迭代 (GPI) 的一般策略，这意味着它们保持了一个近似的值函数和近似的策略，并且它们不断地试图以另一个为基础来改进它们。这三个观点是本书主题的核心。我们认为，价值函数、支持价值更新和 GPI 是强大的组织原则，可能与任何人工或自然的智能模型相关。图 8.11 显示了方法的两个最重要的维度。这些维度与用于改进值函数的更新有关。水平维度是它们是样本更新 (基于样本轨迹) 还是预期更新 (基于可能轨迹的分布)。预期的更新需要一个分布模型，而示例更新只需要一个示例模型，或者可以从没有模型的实际经验中完成 (另一个变化维度)。图 8.11 的垂直维度对应于更新的深度，即自举程度。在这个空间的四个角中有三个是估计值的主要方法: 动态规划、TD 和蒙特卡罗。沿空间的左边缘是样本更新方法，

图 8.11: 通过增强学习方法的空间，突出显示了本书第一部分中探索的两个最重要的维度: 更新的深度和宽度。

从单步 TD 更新到全返回蒙特卡罗更新。这些是一个谱系之间包括方法基于 n-step 更新 (在第 12 章我们将扩展这个混合物 n-step 更新如 -updates 实施资格痕迹)。动态编程方法显

示在空间的右上角，因为它们涉及一步预期更新。右下角是预期更新非常深入的极端情况，它们会一直运行到终端状态 (或者，在一个持续的任务中，直到折扣将任何进一步的奖励降低到可以忽略的程度)。这是彻底搜索的情况。沿着这个维度的中间方法包括启发式搜索和相关方法，这些方法可以搜索和更新到一个有限的深度，可能是有选择性的。也有一些方法是沿着水平维度的中间的。这些包括混合预期更新和示例更新的方法，以及在单个更新中混合示例和分布的方法的可能性。正方形的内部填充了所有这些中间方法的空间。我们在本书中强调的第三个方面是政策方法和政策方法之间的二元区别。在前一种情况下，代理将学习它当前所遵循的策略的值函数，而在后一种情况下，它会学习该策略。

书目的和历史的言论 191

针对不同策略的策略的值函数，通常是代理当前认为最好的策略。由于需要进行探索，策略生成行为通常与当前认为的最佳行为不同。第三个维度可以可视化为垂直于图 8.11 中页面的平面。除了刚才讨论的三个维度之外，我们在整本书中还发现了其他一些维度:

回报的定义是任务是间歇性的还是持续的，折现的还是不折现的? 动作值、状态值、后状态值应该是什么类型的值估计? 如果只估计状态值，那么行为选择需要一个模型或一个单独的策略 (如 actor - critics 方法)。行动选择/探索如何选择行动以确保在探索和开发之间进行适当的权衡? 我们只考虑最简单的方法: -greedy, 乐观的初始化值,soft-max, 上层的信心。

同步与异步是同时执行的所有状态的更新，还是按某种顺序逐个执行的更新?

一个更新是基于真实体验还是模拟体验? 如果两者都有，每一个多少? 更新的位置应该更新哪些状态或状态对? 无模型方法只能在实际遇到的状态和状态操作对之间进行选择，但是基于模型的方法可以任意选择。这里有很多可能性。

更新的时间应该作为选择动作的一部分进行更新，还是只在之后进行更新?

更新的内存应该保留多长时间? 它们是应该永久保留，还是只在计算操作选择时保留，就像在启发式搜索中那样?

当然，这些维度既不是详尽的，也不是相互排斥的。个别算法在许多其他方面也不同，许多算法在几个维度上的几个位置。例如，Dyna 方法使用真实的和模拟的经验来影响相同的值函数。维护以不同方式或不同状态和操作表示方式计算的多个值函数也是完全合理的。然而，这些维度确实构成了一套连贯的概念，用来描述和探索广泛的可能方法。这里没有提到的最重要的维度，也没有在本书的第一部分中介绍，是函数逼近的维度。函数逼近可以看作是一系列可能性的正交谱，从一个极端的列表方法到状态聚合、各种线性方法，再到一组不同的非线性方法。这个维度在第二部分中进行了探讨。

书目的和历史的言论

8.1 在这里提出的规划和学习的总体观点是逐渐发展起来的。多年来，部分由作者 (Sutton, 1990, 1991a, 1991b;Barto, Bradtke, Singh, 1991, 1995; 萨顿和 Pinette,1985; 萨顿和 Barto,1981 b); 它受到 Agre 和 Chapman(1990) 的强烈影响;Agre 1988)，Bertsekas 和 tsiklis (1989)，Singh(1993) 等人。作者还受到潜在学习的心理学研究 (Tolman, 1932) 和对思想本质的心理学观点 (如 Galanter 和 Gerstenhaber, 1956) 的强烈影响;Craik,1943; 坎贝尔,1960; 丹尼特,1978)。在本书的第三部分，第 14.6 节将基于模型的和无模型的方法与学习和行为的心理学理论联系起来，第 15.11 节讨论了大脑如何实施这些方法的想法。

8.2 直接和间接的术语，我们用来描述不同种类的强化学习，来自自适应控制文献 (例如，Goodwin 和 Sin, 1984)，他们被用来做同样的区分。“系统识别”一词用于自适应控制中，我们称之为模型学习 (例如，Goodwin 和 Sin, 1984;Ljung 所以¨derstrom,1983; 年轻,1984)。Dyna 体系结构源自 Sutton(1990)，本节和下一节的结果基于报告的结果。Barto 和 Singh(1990) 在比较直接和间接强化学习方法时考虑了一些问题。早期的工作扩展强啡肽线性函数近似 (第 9 章) 是由 Sutton Szepesvári,Geramifard, 和保龄球 (2008) 和帕尔, 李泰勒, 李曼荣 Painter-Wakefield,(2008)。

8.3 基于模型的强化学习已经有了一些成果。探索加值和乐观初始化的思想达到了逻辑的极限，其中所有未完全探索的选择都假定为最大回报，并计算出最优路径来测试它们。Kearns 和 Singh(2002) 的 E3 算法和 Brafman 和 Tennenholtz(2003) 的 R-max 算法保证在状态数和

动作数的时间多项式中找到一个接近最优的解。对于实际的算法来说，这通常太慢，但在最坏的情况下，这可能是最好的。

8.4 优先级清扫是由 Moore 同时独立开发的还有阿特克森 (1993)，彭和威廉姆斯 (1993)。第 170 页的方框中的结果是由 Peng 和 Williams(1993) 提出的。第 171 页方框中的结果是由摩尔和阿特克森得出的。这一领域的主要后续工作包括 McMahan 和 Gordon(2005) 以及 van Seijen 和 Sutton(2013)。

8.5 这一节受到 Singh(1993) 实验的强烈影响。

8.6-7 轨迹采样是增强学习的一部分 Barto、Bradtke 和 Singh(1995) 在介绍 RTDP 时，最明确地强调了这一点。他们认识到 Korf(1990) 的学习

书目的和历史的言论 193 年

实时 A* (LRTA*) 算法是一种适用于随机问题以及 Korf 关注的确定性问题的异步 DP 算法。除了 LRTA* 之外，RTDP 还包括在执行操作之间的时间间隔内更新许多状态值的选项。Barto 等人 (1995) 通过结合 Korf(1990) 收敛证明 LRTA* 与 Bertsekas (1982) (Bertsekas 和 Tsitsiklis, 1989) 的结果，证明了这里所描述的收敛结果，确保了异步 DP 在未折现情况下的随机最短路径问题的收敛性。将模型学习与 RTDP 结合称为适应性 RTDP, Barto 等人 (1995) 也提出了这一观点，Barto(2011) 也对此进行了讨论。

8.9 为了进一步阅读启发式搜索，鼓励读者查阅课文。罗素和诺维格 (2009) 和科尔夫 (1988) 的调查。Peng 和 Williams(1993) 研究了一个前沿的更新焦点，就像本节所建议的那样。

8.10 Abramson(1990) 提出的期望结果模型是一种应用于 2 -的推出算法人的游戏，两个模拟玩家的游戏都是随机的。他认为，即使是随机游戏，它也是一种“强大的启发式”，“精确、准确、易于估计、有效计算和领域无关”。Tesauro 和 Galperin(1997) 通过使用不同随机生成的掷骰序列来计算西洋双陆棋的位置，采用了“滚滚出”这个术语，证明了滚出算法在改进西洋双陆棋程序的表现上的有效性。Bertsekas、tsiklis 和 Wu(1997) 研究了应用于组合优化问题的推出算法，Bertsekas(2013) 调查了它们在离散确定性优化问题中的应用，并指出它们“通常非常有效”。

8.11 Coulom(2006) 和 Kocsis 介绍了 MCTS 的核心思想和 Szepesvári(2006)。他们在之前的研究中建立了蒙特卡罗规划算法。Browne, Powley, Whitehouse, Lucas, Cowling, Rohlfshagen, Tavener, Perez, Samothrakis 和 Colton(2012) 是对 MCTS 方法及其应用的极好的调查。大卫 • 西尔弗对这一节的观点和介绍做出了贡献。

Part II

近似解的方法

在本书的第二部分，我们扩展了在第一部分中提出的表格方法来应用于任意大的状态空间的问题。在我们想要应用强化学习的许多任务中，状态空间是组合的和巨大的; 例如，可能的相机图像的数量要比宇宙中原子的数量大得多。在这种情况下，即使在无限的时间和数据的限制下，我们也不能期望找到最优策略或最优值函数; 我们的目标是利用有限的计算资源找到一个好的近似解。在本书的这一部分，我们探讨了这种近似解的方法。大状态空间的问题不只是大表所需的内存，而是精确填充它们所需的时间和数据。在我们的许多目标任务中，几乎所有遇到的状态以前都不会出现。要在这种状态下作出明智的决定，必须从与不同状态的接触中归纳出与当前状态在某种意义上相似的情况。换句话说，关键问题是泛化。如何能对状态空间的有限子集进行有效的推广，从而在更大的子集上产生良好的近似? 幸运的是，我们已经广泛地研究了从实例中归纳，我们不需要发明全新的方法用于强化学习。在某种程度上，我们只需要将强化学习方法与现有的泛化方法结合起来。我们需要的那种泛化通常被称为函数逼近，因为它从一个期望的函数 (例如，一个值函数) 中抽取例子，并试图从它们中进行归纳，以构建整个函数的近似。函数逼近是一种监督学习的实例，是机器学习、人工神经网络、模式识别和统计曲线拟合的主要研究课题。理论上，在这些领域中研究的任何方法都可以在增强学习算法中充当函数逼近者的角色，尽管在实践中有些方法比其他方法更容易适应这个角色。函数逼近强化学习涉及到许多常规监督学习中不常见的新问题，如非平稳性、自举和延迟目标。我们先后在这部分的五章中介绍了这些问题和其他问题。首先，我们限制了对政策培训的关注，在第 9 章中，我们处理了预测案例，在这个案例中，政策是给定的，只有它的值函数是近似的，然后在第 10 章中，控制案例中，我们找到了最优政策的一个近似。在第 11 章中讨论了带函数逼近的非策略学习问题。在这三章中，我们将必须回到第一原则，重新审视学习的目标，并考虑函数的近似。第 12 章介绍并分析了资格跟踪的算法机制，极大地提高了多步强化学习方法在许多情况下的计算性能。本部分的最后一章探讨了一种不同的控制方法——策略梯度法，它直接逼近最优策略，不需要形成一个近似的值函数 (尽管如果它们也近似于一个值函数，那么它们可能会更有效)。

9. 第九章在政策与近似预测

在这一章, 我们开始我们的研究强化学习的函数近似考虑其使用在评估州值函数从政策数据, 也就是说, 在近似 v 从经验中使用一个已知的生成策略 。新奇的在这一章是近似值函数而不是表表示为一个参数化函数形式与权向量 $w\ Rd$。我们将编写 $\hat{v}(s,w)\ v\ (s)$ 的近似值状态 s 给定的权向量 w。例如,$\hat{v}$ 可能是一个线性函数的功能状态, 与 w 权重向量的特性。更普遍的是,$\hat{v}$ 可能由多层人工神经网络函数计算, 与 w 所有层的连接权值向量。通过调整权重，网络可以实现各种不同的功能。或 $\hat{v}$ 可能函数计算一个决策树, 在 w 是所有数字定义分割点和叶树的值。通常，权值的数量 (w 的维数) 要比状态数 (d? 而改变一个权重就会改变许多状态的估计值。因此，当一个状态被更新时，更改将从该状态推广到影响许多其他状态的值。这样的泛化使学习可能更强大，但也可能更难于管理和理解。也许令人惊讶的是，将强化学习扩展到函数逼近也使它适用于部分可观察的问题，在这种问题中，代理无法获得完整的状态。如果参数化函数形式 $\hat{v}$ 不允许估计价值取决于国家的某些方面, 那么它同样如果这些方面是不可见的。事实上，书中这部分给出的函数逼近方法的所有理论结果都同样适用于局部可观测的情况。然而，函数逼近所不能做的是用过去观察的记忆来增加状态表示。第 *17.3* 节简要讨论了一些可能的进一步扩展。

9.1 值函数逼近

本书所介绍的所有预测方法都被描述为对估计值函数的更新，该函数在特定状态下将其值转换为该状态的“备份值”或更新目标。让我们参考个体更新到符号年代?→u,s 是状态更新和 u 是年代的估计价值的更新目标转向。例如, 蒙特卡洛更新值预测是圣?→Gt,TD(0) 更新是圣?→Rt + 1 + v̂(wt) 圣 + 1, 圣是 n-step TD 更新?→Gt:t + n。DP(动态规划) 政策评估更新,s ?→E (Rt + 1 + v̂(wt 圣 + 1)| 圣 = s], 任意状态更新, 而在其他情况下国家遇到的实际经验, 圣, 更新。将每次更新解释为指定值函数所需的输入-输出行为的示例是很自然的。在某种意义上, 更新年代?→u 意味着国家年代估计价值应该更像更新目标 u。到目前为止, 实际的更新一直微不足道: 年代的估计价值的表条目只是一小部分的方式转向你, 和所有其他国家的估计价值保持不变。现在，我们允许任意复杂和复杂的方法来实现更新，并在 s 处进行更新，以使许多其他状态的估计值也被更改。以这种方式学习模拟输入输出示例的机器学习方法称为监督学习方法，当输出是数字时，如 u，这个过程通常称为函数逼近。函数逼近方法期望得到它们试图逼近的函数的预期输入-输出行为的例子。我们用这些方法来预测价值仅仅通过他们的年代?→g 的每个更新作为一个培训的例子。然后我们将它们产生的近似函数解释为一个估计值函数。以这种方式将每个更新看作常规的训练示例，使我们能够使用各种现有的函数逼近方法中的任何一种进行值预测。原则上，我们可以用任何方法来监督学习，包括人工神经网络、决策树和各种各样的多元回归。然而，并不是所有的函数逼近方法都同样适用于强化学习。最复杂的人工神经网络和统计方法都假设有一个静态训练集，在此基础上进行多次

传递。然而，在强化学习中，重要的是学习能够在线进行，同时代理与它的环境或它的环境模型进行交互。要做到这一点，需要能够有效地从增量获得的数据中学习的方法。此外，增强学习通常需要函数逼近方法来处理非平稳目标函数 (随着时间变化的目标函数)。例如, 在控制方法基于谷歌价格指数 (广义政策迭代) 我们经常寻求学习 q 的变化。即使策略保持不变，如果训练示例的目标值是通过引导方法 (DP 和 TD 学习) 生成的，那么它们也是非平稳的。不容易处理这种非平稳性的方法不太适合强化学习。

9.2 预测目标 (VE)

到目前为止，我们还没有明确明确的预测目标。在表格式的情况下，不需要连续的预测质量，因为学习的值函数可以精确地等于真正的值函数。此外，每个状态的学习值都是解耦的——一个状态的更新不影响其他状态。但是用真正的近似值，一个状态的更新会影响许多其他状态，并且不可能使所有状态的值都完全正确。假设我们有更多的状态，而不是权重，所以让一个状态的估计值更准确必然意味着让另一个状态的估计值更不准确。我们有义务说出我们最关心的国家。我们必须指定一个国家分布 (s) 0 ? s (s)= 1, 表示我们关心多少在每个州年代错误。错误的年代我们意味着区别近似值 v 的平方ˆ(s,w) 和真正价值 v (年代)。权重这个的状态空间, 我们获得一个天然的目标函数, 均方值错误, 已经表示为:

这个度量的平方根，也就是根 VE，给出了一个粗略的度量，来表示近似值与真实值之间的差异，并且通常在绘图中使用。选择经常 (s) 的一部分时间在 s。在政策培训这叫做政策分布; 在这一章中，我们完全关注这个案例。在持续的任务, 在政策分布是 下的平稳分布。

这两种情况，连续的和偶发的，表现相似，但近似地，它们必须在形式分析中单独处理，我们将在本书的这一部分反复看到。这就完成了学习目标的说明。但对于强化学习来说，VE 是否是正确的性能目标还不完全清楚。记住，我们的终极目标——我们学习价值功能的原因——是找到更好的政策。为此目的的最佳值函数不一定是最小化 VE 的最佳值函数。然而，目前还不清楚价值预测的一个更有用的替代目标是什么。现在，我们将关注 VE。理想目标而言, 已经是找到全局最优, 权重向量 w∗ 为 VE(w∗) (w) 所有可能的 w。达到这个目标等简单的函数近似者有时可能是线性的, 但很少有可能复杂函数近似者, 如人工神经网络和决策树。短, 复杂的函数近似者可能寻求而不是局部最优, 收敛的权向量 w∗VE(w∗) VE(w)w w∗ 的一些社区。虽然这种保证只是稍微让人放心，但对于非线性函数近似器来说，它通常是最好的，而且通常已经足够了。尽管如此，对于许多对强化学习感兴趣的情况来说，并不能保证收敛到一个最优值，甚至在一个最优值的有限距离内。有些方法实际上是有分歧的，因为它们的极限接近无穷。在前两部分中，我们概述了一个框架，用于将广泛的增强学习方法与广泛的函数逼近方法结合起来，使用前者的更新为后者生成训练示例。我们还描述了一个 VE 性能度量，这些方法可能希望最小化它。可能的函数近似方法的范围太大了，无法涵盖所有的内容，而且无论如何，大多数人都不知道如何进行可靠的评估或推荐。当然，我们只考虑一些可能性。在本章的其余部分，我们将重点讨论基于梯度原理的函数逼近方法，特别是线性梯度下降法。我们关注这些方法，部分是因为我们认为它们特别有前途，也因为它们揭示了关键的理论问题，但也因为它们很简单，我们的空间有限。

9.3 随机梯度和半梯度方法

本文详细介绍了一类基于随机梯度下降 (SGD) 的函数逼近方法。SGD 方法是所有函数逼近方法中应用最广泛的一种，特别适合在线强化学习。在梯度下降法中，权向量是具有固定数目实值分量 w 的列向量。= (w1, w2，⋯),wd)?,1 和近似值函数 v̂(s,w) 是一个可微函数的 w s s . 我们将更新 w 在一系列离散时间的每个步骤,t = 0,1,2,3,。, 所以我们需要一个符号的 wt

1 的? 表示转置，这里需要将文本中的水平行向量转换为垂直列向量; 在这本书中，向量通常被认为是列向量，除非显式地写出水平或转置。每一步的权向量。现在, 让我们假定, 在

每一步, 我们观察一个新的例子圣?→v (St) 组成的 (可能是随机选择的) 州圣和其真正价值的政策。这些状态可能是与环境交互的连续状态，但目前我们不这样认为。即使我们给出精确的, 正确的价值观,v 每个圣 (St), 还有一个困难的问题, 因为我们的函数近似者资源有限, 因此有限的决议。特别地，通常没有 w 能得到所有的状态，甚至所有的例子，完全正确。此外，我们必须对没有在例子中出现的所有其他国家一概而论。我们假设状态出现在示例相同的分布, , 我们试图最小化由 (9.1)。在这种情况下，一个好的策略是尽量减少所观察到的示例上的错误。随机梯度下降 (SGD) 方法通过在每个示例之后对权重向量进行少量的调整，从而最大程度地减少该示例上的错误: 是一个积极的步长参数, f(w), 为任何标量表达式 f(w), 是一个向量的函数 (w), 表示列向量的偏导数表达式对向量的分量: f(w)。=

w1 f(w) , f(w) w2 ,。, f wd(w)

. (9.6)

这个导数向量是 f 对 w 的梯度，SGD 方法是“梯度下降”方法，因为在 wt 中，整体的步长与例子的平方误差的负梯度成正比 (9.4)。这是误差下降最快的方向。当更新完成时，梯度下降方法被称为“随机”，这里只对一个示例进行更新，这个示例可能是随机选择的。在许多示例中，进行小步骤，总体效果是最小化平均性能度量，如 VE。为什么 SGD 在梯度方向上只走了一小步，这一点可能不会马上显现出来。我们能不能沿着这个方向移动并完全消除这个例子中的错误? 在许多情况下，这是可以做到的，但通常是不可取的。请记住，我们并不是在寻找或期望找到一个对所有状态都有零误差的值函数，而只是一个平衡不同状态下误差的近似。如果我们在一步中完全纠正了每个例子，那么我们就不会找到这样的平衡。事实上, 收敛结果 SGD 假定 随时间的方法。如果以满足标准随机逼近条件 (2.7) 的方式减小，则保证 SGD 方法 (9.5) 收敛到局部最优。我们现在的目标输出, 这里表示 Ut R,t 的训练示例中, 圣?→Ut, 不是真正价值,v (圣), 但一些, 可能是随机的, 近似。例如,Ut 可能的系统版本的 v (圣), 也可能是一个引导使用 v̂ 在前一节中提到的目标。在这些情况我们不能执行的更新 (9.5) 因为 v (St) 是未知的, 但是我们可以把它近似取代 Ut 代替 v (St)。这就产生了以下状态值预测的通用 SGD 方法:

wt + 1 . = wt +

Ut−v̂(圣 wt)

wt v̂(St)。(9.7)

如果 Ut 是一个无偏估计, 也就是说, 如果 E[Ut | 圣 = s]= v (圣), 对于每一个 t, 然后 wt 保证收敛于局部最优通常随机近似条件下降低 (2.7)。例如, 假设美国的例子是美国所产生的交互 (或模拟交互) 与环境使用政策 。因为一个状态的真实值是它之后的收益的期望值，蒙特卡罗目标 Ut。= Gt 是通过定义一个无偏估计的 v (St)。有了这个选择, 一般 SGD 方法 (9.7) 收敛于局部最优近似 v (St)。从而保证了蒙特卡罗状态值预测的梯度下降版本能够找到局部最优解。完整算法的伪代码如下所示。

不获得相同的担保, 如果引导估计 v (St) 作为目标 Ut(9.7)。引导目标，如 n 步返回 Gt:t+n 或 DP 目标

一个年代?,r (| St)p(s ?r | 圣,)[r + v̂(s ? wt)] 所有依赖的当前值重量矢量 wt，这意味着它们会有偏差，不会产生真正的梯度下降法。看这个的一个方法是, 关键步骤从 (9.4)、(9.5) 依赖于目标独立 wt。这一步不会有效如果引导估计是用于代替 v (St)。引导方法实际上不是真正的梯度下降的实例 (Barnard, 1993)。它们考虑了改变权重向量 wt 对估计的影响，但忽略了它对目标的影响。它们只包含梯度的一部分，因此，我们称之为半梯度法。虽然半梯度 (bootstrapping) 方法不像梯度方法那样强收敛，但它们在重要的情况下确实可靠地收敛，如下一节讨论的线性情况。此外，他们还提供了一些重要的优势，这些优势使他们更喜欢。原因之一是，它们通常能显著提高学习速度，正如我们在第 6 章和第 7 章中看到的那样。另一个原因是他们能够学习。

持续在线，不要等待一集的结束。这使它们能够用于持续的问题，并提供计算优势。一个典型的半梯度方法是半梯度 TD(0)，它使用 Ut。= Rt + 1 + v̂(圣 + 1 w) 作为它的目标。该方法的完整伪代码在下面的框中给出。状态聚合是一种简单的推广函数逼近形式，其中状态

被分组在一起，每个组有一个估计值 (权向量 w 的一个分量)。状态的值被估计为其组的组件，当状态被更新时，只更新该组件。国家 SGD 聚合是一种特殊情况 (9.7) 的梯度, v̂(St,wt), 为圣的组件是 1 和 0 的其他组件。示例 9.1:1000 状态随机游走上的状态聚合考虑一个 1000 状态的随机游走任务 (第 125 和 144 页上的示例 6.2 和 7.1)。这些州的编号从 1 到 1000，从左到右，所有的剧集都在 500 州的中心开始。状态转换是从当前状态到它左边的 100 个相邻状态之一，或者到它右边的 100 个相邻状态之一，所有的概率都是相等的。当然，如果当前状态接近边缘，那么在它的一侧可能会有少于 100 个邻居。在这种情况下，所有消失的邻居的概率都变成了在这一边终止的概率 (因此，状态 1 有 0.5 的概率在左边终止，状态 950 有 0.25 的概率在右边终止)。像往常一样, 左边终止产生的奖励 −1, 右边和终止生产 + 1 的奖励。所有其他的转变都是零奖励。我们将此任务作为贯穿本节的运行示例。图 9.1 显示了真值函数 v 这项任务。它几乎是一条直线，但在每一端的最后 100 个状态中，它稍微向水平方向弯曲。也显示最后学到的近似值函数梯度蒙特卡罗算法与国家聚合后 100000 集的步长 = 2×10−5。对于状态聚合，1000 个状态被划分为 10 组，每组 100 个状态 (即美国的州 1-100 是一个群体，州 101-200 是另一个群体，以此类推。楼梯的效果价值尺度近似 MC v̂ 价值分布范围 0 0 国家 1000 图 9.1 在 1000 态随机漫步任务中，状态聚合的函数逼近，使用梯度蒙特卡罗算法 (第 202 页)。图中所示为典型的状态聚合; 在每一组中，近似值都是常数，并且从一组突然变化到另一组。这些近似值接近 VE(9.1) 的全局最小值。一些细节的近似值是最好的赞赏, 参照国家分布 这个任务, 较低部分的图所示右规模。位于中心的状态 500 是每一集的第一个状态，但很少再次访问。平均来说，大约 1.37% 的时间步骤花费在开始状态。从一开始就可到达的状态是第二次访问，在每一个步骤中花费了 0.17% 的时间。从那里几乎线性 脱落, 达到极端状态 1 和 1000 年的 0.0147% 左右。最明显的分布影响是在最左边的组，其值明显地比组内状态的真实值的未加权平均值高，而在最右边的组，其值明显地降低。这是由于美国在这些领域拥有他们的权重最大的不对称 。例如，在最左边的组中，状态 100 的权重是状态 1 的 3 倍。因此，对组的估计偏向于状态 100 的真实值，它高于状态 1 的真实值。

9.4 线性方法

最重要的一个特殊情况的近似函数, 近似函数,v̂(• w), 是一个线性函数的权向量,w。对应于每一个州, 有一个实值向量 x(年代)。= (x1(s)，x2(s)，⋯,xd(s))? 与 w 的分量相同，线性方法近似于状态值函数。

w 和 x 之间的内积 (s):v̂(s,w)。= w ? x(s)。= d ?i = 1 wixi(年代)。(9.8)

在这种情况下，近似值函数在权值中是线性的，或者仅仅是线性的。向量 x(s) 被称为特征向量代表国家。每个组件 xi(s)x(s) 的值是一个函数 xi:s→r . 我们认为一个功能完整的其中一个功能, 我们调用它的价值特性的年代。线性方法, 特征基函数, 因为它们形成一个近似的线性基础设置功能。构造表示状态的 d 维特征向量与选择一组 d 基函数是一样的。特性可以用许多不同的方式定义; 我们将在下一节中讨论一些可能性。使用线性函数近似的 SGD 更新是很自然的。在这种情况下，近似函数关于 w 的梯度是

v̂(s,w)= x(年代)。因此，在线性情况下，一般 SGD 更新 (9.7) 简化为一种特别简单的形式:

wt + 1 . = wt +

Ut−v̂(圣 wt)

x(St)。因为线性 SGD 是如此简单，所以它是最适合进行数学分析的。几乎所有用于学习各种系统的有用的收敛结果都适用于线性 (或更简单的) 函数逼近方法。特别是，在线性情况下，只有一个最优解 (或者在退化情况下，有一组同样好的最优解)，因此，任何保证收敛到局部最优解或接近局部最优解的方法都自动保证收敛到全局最优解或接近全局最优解。例如, 在前一节中给出的梯度蒙特卡罗算法收敛于全局最优线性函数近似下的 VE 如果 随时间减少根据通常的条件。上一节介绍的半梯度 TD(0) 算法也在线性函数近似下收敛，但这并不符

合 SGD 的一般结果; 一个单独的定理是必要的。收敛到的权向量也不是全局最优，而是接近局部最优的点。更详细地考虑这个重要的情况是有用的，特别是对于持续的情况。每次 t 的更新是

wt + 1 . = wt +

Rt + 1 + w 吗?t xt + 1−w ?t xt

xt (9.9)

= wt +

Rt + 1 xt−xt

xt− xt + 1

wt

,

这里我们用了符号简写 xt = x(St) 一旦系统达到稳定状态，对于任何给定的 wt，都可以写出期望的下一个权重向量

E(wt + 1 | wt)= wt + (b−Awt) (9.10)

在哪里

b。= E(Rt + 1 xt) Rd 和 A。= E

xt

xt− xt + 1

Rd×Rd(9.11) 由 (9.10) 可知，如果系统收敛，则它必须收敛于权向量 wTD

b−AwTD = 0 b = AwTD 西医. =−1 b。(9.12) 这个量叫做 TD 定点。事实上，线性半梯度 TD(0) 收敛到这一点。盒子里给出了证明它收敛性的一些理论，以及上面的逆的存在性。

在 TD 的不动点，它也被证明 (在继续的情况下)，VE 在最小可能误差的有界扩展范围内:

即渐近 TD 方法误差不超过 1− * 尽可能最小的误差, 达到在蒙特卡罗方法的限制。因为通常是附近的一个, 这个扩展因数可能很大, 所以有大量潜在损失与 TD 渐近性能的方法。另一方面，回想一下，TD 方法通常比蒙特卡罗方法的方差大得多，因此更快，正如我们在第 6 章和第 7 章所看到的那样。哪种方法最好取决于近似和问题的性质，以及学习的时间。

类似于 (9.14) 的约束也适用于其他策略上的引导方法。例如，线性半梯度 DP (Eq. 9.7 with Ut)。= ? (圣)|

s?,r p(s ?r | 圣,)[r + v̂(s ? wt)]) 与更新根据政策分布也将收敛于 TD 定点。一步半梯度动作值方法，如下一章所涉及的半梯度 Sarsa(0)，收敛到一个类似的不动点和一个类似的界。对于情景性任务，有一个稍微不同但相关的界限 (参见 Bertsekas 和 tsiklis, 1996)。在奖励、特性和降低步长参数方面也有一些技术条件，我们在这里省略了这些。完整的细节可以在原文中找到 (Tsitsiklis 和 Van Roy, 1997)。对这些收敛结果至关重要的是，根据策略分布更新状态。对于其他更新分布，使用函数逼近的自举方法实际上可能会发散到无穷大。第 11 章给出了这方面的例子和可能的解决方法的讨论。例 9.2: 对 1000 状态随机游走状态聚合的自举是线性函数逼近的一个特例，因此让我们回到 1000 状态随机游走来说明本章的一些观察结果。图 9.2 的左面板显示了使用相同的状态聚合 (如示例 9.1) 所学习的半梯度 TD(0) 算法 (page 203) 所学习的最终值函数。我们看到，与图 9.1 所示的蒙特卡罗逼近相比，接近渐近的 TD 逼近确实距离真实值更远。尽管如此，TD 方法在学习速率方面仍有很大的潜在优势，并且在第 7 章中，我们用 n-step TD 方法对蒙特卡罗方法进行了全面的研究。图 9.2 右边的面板显示了使用 n 阶半梯度 TD 方法的结果，该方法使用了 1000 状态随机漫步上的状态聚合，这与我们之前用表格方法和 19 状态随机漫步得到的结果非常相似 (图 7.2)。为了获得定量相似的结果，我们将状态聚合转换为 20 组，每组 50 个状态。然后，这 20 组在数量上接近

0.5

0.45

0.35 - 0.4 平均均方根误差超过 1000 个州和前 10 个州。集

0.3

0.25

图 9.2: 在 1000 状态随机游走任务上使用状态聚合进行引导。左: 半梯度 TD 的渐近值要比图 9.1 中的渐近蒙特卡罗值差。正确: 具有状态聚合的 n 步方法的性能与具有表格表示的方法非常相似 (cf. 图 7.2)。这些数据是超过 100 次的平均值。

到表格问题的 19 个状态。特别是，回想一下，状态转换在左边或右边最多有 100 个状态。一个典型的跃迁将是向右或向左 50 个状态，这在数量上类似于 19 个状态列表系统的单状态跃迁。为了完成匹配，我们在这里使用相同的性能度量——对所有状态和前 10 集的 RMS 误差的非加权平均——而不是使用函数逼近时更合适的 VE 目标。

上面示例中使用的半梯度 n-step TD 算法是第 7 章提出的表格 n-step TD 算法对半梯度函数逼近的自然扩展。伪代码在下面的框中给出。

该算法的关键方程类似于 (7.2)

wt + n 。= t + n−1 + (Gt:t + n−v̂(圣、wt + n−1)] v̂(St,wt + n−1),0 t < t(9.15)

n 阶返回从 (7.1) 推广到哪里

Gt:t + n。= Rt + 1 + Rt + 2 + • • • + n−1 Rt + n + nv̂(wt 圣 + n + n−1),0 t t−n。(9.16) 练习 9.1 显示，如本书第一部分所介绍的表列方法是线性函数逼近的一个特例。特征向量是什么??

210 年第 9 章: 带有近似的政策预测

9.5 线性方法的特征构造

线性方法之所以有趣，是因为它们的收敛性保证，但也因为在实践中，它们在数据和计算方面都是非常有效的。无论这是否如此，关键取决于各州如何以特征来表现，我们在这一大块区域进行调查。选择适合于任务的特性是向强化学习系统中添加先验域知识的重要方法。直觉上，这些特征应该对应于状态空间的各个方面，在这些方面可以适当地进行泛化。例如，如果我们对几何对象进行估值，我们可能想要为每个可能的形状、颜色、大小或函数提供特征。如果我们评估一个移动机器人的状态，那么我们可能想要有关于位置、剩余电量的程度、最近的声纳读数等等的特性。线性形式的局限性是它不能考虑任何之间的交互功能, 如功能的存在我好只是在缺乏特性 j。例如, 在 pole-balancing 任务 (例如 3.4) 高角速度可以是好是坏取决于角度。如果角度很高，那么高的角速度意味着即将下降的危险——一种糟糕的状态——而如果角度很低，那么高的角速度意味着极点正在调整自身——一种良好的状态。如果一个线性值函数的特征分别表示角度和角速度，它就不能表示这个。相反，它需要这两个底层状态维度的组合的特性。在下面的小节中，我们将考虑各种通用的方法。

9.5.1 多项式

许多问题的状态最初被表示为数字，例如在平衡的任务中的位置和速度 (例 3.4)，在杰克的汽车租赁问题中的每一辆车的数量 (例 4.2)，或赌徒的资本在赌徒问题 (例 4.3)。在这些类型的问题中，函数逼近增强学习与我们熟悉的插值和回归任务有很多共同之处。通常用于插补和回归的各种特征族也可用于强化学习。多项式是用于插值和回归的最简单的特征族之一。虽然我们在这里讨论的基本多项式特征在强化学习中并不像其他类型的特征那样有效，但它们作为一个很好的介绍，因为它们简单而熟悉。作为一个例子，假设一个强化学习问题有两个数值维度的状态。一个代表国家, 让它的两个数字是 s1 R 和 s2 R . 您可以选择代表年代仅仅通过两个州的维度, 因此 x(s)=(s1,s2)? 但是，你将不能考虑这些维度之间的任何相互作用。此外，如果 s1 和 s2 都为 0，那么近似值也必须为 0。这两个限制都可以通过用四维特征向量 x(s) = (1,s1, s2, s1s2) 来表示 s 来克服。最初的 1 特性允许在原始状态数中表示仿射函数，而最终的产品特性 s1s2 则允许考虑交互。或者你可以选择使用高维的特征向量比如 x(s) = (1 s1s2 s2 s1s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 s2 来

9.5.2 傅里叶基础

另一种线性函数近似方法基于傅立叶级数，它将周期函数表示为不同频率的正弦和余弦基函数的加权和 (特征)。(一个函数是周期如果 f(x)= f(x +) 对于所有 x 和一些时间 。) 傅里叶级数和更一般的傅里叶变换广泛应用于应用科学, 部分是因为如果一个函数近似, 然后给出了基函数的权重通过简单的公式, 此外, 有足够的基函数本质上任何函数可以近似为准确。在强化学习中，当要逼近的函数未知时，傅里叶基函数很有趣，因为它们易于使用，并且在一系列强化学习问题中表现良好。首先考虑一维情况。通常函数的傅里叶级数表示的一维时间 表示函数的线性组合正弦和余弦函数均匀划分的每个周期与时间 (换句话说, 其频率是基频的整数倍 1 /)。但是如果你感兴趣的近似一个非周期函数定义在有界区间, 然后您可以使用这些傅里叶基础特性与 区间长度。有趣的函数就是正弦函数和余弦函数的周期线性组合。此外, 如果您设置 间隔长度的两倍利息和限制注意近似半区间 [0, / 2], 然后你可以用余弦特性。这是可能的，因为你可以表示任何偶函数，也就是说，任何关于原点对称的函数，只有余弦基。所以任何函数的半周期 (0, / 2) 可以近似为密切的有足够的余弦特性。(说“任何函数”都不是完全正确的，因为函数必须在数学上表现良好，但这里我们跳过了这个技术性问题。) 或者，也可以只使用正弦函数的特征，它们的线性组合总是奇函数，也就是关于原点的不对称函数。但是最好只保留余弦函数，因为“半偶数”函数比“半奇数”函数更容易近似，因为后者在原点处通常是不连续的。当然, 这并不排除使用正弦和余弦特性近似在区间 [0, / 2], 在某些情况下这可能是有利的。按照这一逻辑, 让 = 2, 这样定义的特点是 half- 区间 [0,1], 一维傅里叶余弦 n 基础由 n + 1 的特性

习近平 (s)= cos(i s) 年代 [0,1], 对于 i = 0，…图 9.3 给出了一维傅里叶余弦的特征 xi，对于 i = 1,2,3,4;x0 是一个常数函数。

图 9.3: 一维傅里叶余弦基特征 xi, i = 1,2,3,4，用于在区间 [0,1] 上逼近函数。在 Konidaris 等 (2011) 之后。

这种推理同样适用于多维情况下的傅立叶余弦级数逼近，如下面的框所示。

产品年代?ci 具有在 0，…在一维情况下，这个整数决定了特征沿该维度的频率。这些特性当然可以根据特定应用程序的有界状态空间进行移动和缩放。

以 k = 2 为例，其中 s = (s1, s2)?，其中每个 ci = (ci 1, ci 2)? 图 9.4 显示了六个傅立叶余弦特征的选择，每个特征都由定义它的向量 ci 标记 (s1 是水平轴，ci 显示为一个行向量，索引我省略了)。任何在 c 中的零意味着这个特征在状态维上是不变的。如果 c = (0,0)，特征在两个维度上都是常数; 如果 c = (c1, 0)? 该特征在第二个维度上是常数，在第一个维度上随着频率的变化而变化，取决于 c1; 类似地，对于 c = (0,c2)? 当 c = (c1, c2) 时? 当 cj 都不为 0 时，特征在两个维度上都不同，表示两个状态变量之间的交互。c1 和 c2 的值决定了每个维度上的频率，它们的比值给出了交互的方向。

1

图 9.4:6 个二维傅立叶余弦特征的选择，每个特征都由定义它的向量 ci 标记 (s1 是水平轴，ci 用索引 i 省略)。在 Konidaris 等 (2011) 之后。

在学习算法 (9.7)、半梯度 TD(0) 或半梯度 Sarsa 中使用傅里叶余弦特征时，可能需要对每个特征使用不同的步长参数。如果基本的步长参数 , 然后 Konidaris Osentoski, 和托马斯 (2011) 建议设置步长参数特性 xi i = / ?(ci 1)2 + ••• + k(ci)2(除非每个 ci j = 0, 在这种情况下 i =)。傅立叶余弦特征与 Sarsa 相比可以产生良好的性能

基函数的其他集合，包括多项式和径向基函数。然而，并不奇怪的是，傅里叶特征在不连续上有问题，因为除非包含非常高的频率基函数，否则很难避免在不连续点周围“环绕”。n 阶傅里叶基础的数量特征与状态空间的维数呈指数级增长, 但如果这尺寸足够小 (例如,k 5), 可以选择一个 n, 然后可以使用所有的 n 阶傅里叶特征。这使得特性的选择几乎是自动的。然而，对于高维状态空间，有必要选择这些特性的子集。这可以用先验的关于函数性质的信念来完成，并且一些自动化的选择方法可以适应增强学习的增量和非固定的性质。傅里叶基础特性在这方面的一个优势是, 它很容易选择特征向量通过设置 ci 占疑似状态变量之间的相互

作用和通过限制 cj 向量中的值, 这样近似就可以滤除高频组件被认为是噪音。另一方面，由于傅里叶特性在整个状态空间上是非零的 (除了少数几个零之外)，它们代表了状态的全局属性，这使得找到表示本地属性的好方法变得困难。图 9.5 显示了在 1000 态随机漫步示例上比较傅里叶和多项式基础的学习曲线。一般来说，我们不推荐使用多项式进行在线学习

. 3

已经平均 30 多个运行 0 0 5000 年集

图 9.5: 在 1000 状态随机游走时的傅立叶基与多项式。所示为梯度蒙特卡罗方法的学习曲线，该方法具有阶为 5、10 和 20 的傅里叶和多项式基。每种情况下的优化步长参数大致: = 0.0001 的多项式基础和 = 0.00005 的傅里叶。性能度量 (y 轴) 是均方根值误差 (9.1)。

有一些多项式的族比我们讨论过的更复杂，例如，不同的正交多项式的族，这些可能更好的工作，但目前很少有经验与它们在强化学习。

9.5.3 粗编码

图 9.6: 粗编码。Generaliza - 从状态 s 到状态 s? 取决于他们的特征的数量位域 (在本例中是圆) 重叠。这些状态有一个共同点，所以它们之间会有一些泛化。考虑一个任务，其中状态集的自然表示是一个连续的二维空间。这种情况的一种表示是由状态空间中与圆对应的特征组成的，如图右所示。如果状态在一个圆内，则对应的特征值为 1，并表示存在; 否则，该特性为 0，并表示不存在。这种 1 - 0 值的特性称为二进制特性。给定一个状态，其中的二进制特征表示状态所在的圆圈，因此对其位置进行粗码。以这种方式表示具有重叠特性的状态 (尽管它们不必是圆或二进制) 称为粗编码。假设线性梯度下降函数近似，考虑圆的大小和密度的影响。每个圈对应的是受学习影响的单个权重 (w 的一个分量)。如果我们在一个状态下训练，空间中的一个点，那么所有圆的权值相交的状态将会受到影响。因此，由 (9.8) 可知，近似值函数在圆的联盟内的所有状态都会受到影响，一个点与状态“共同”的圆越多，其影响越大，如图 9.6 所示。如果圆圈很小，那么泛化将超过一段很短的距离，如图 9.7(左) 所示，而如果圆圈很大，泛化将超过一段很长的距离，如图 9.7(中) 所示。此外,

图 9.7 线性函数逼近方法的概化特征接受域的大小和形状。这三种情况大致相同特征的数量和密度。

特征的形状将决定泛化的性质。例如，如果它们不是严格的圆形，而是在一个方向上拉长，那么泛化也会受到类似的影响，如图 9.7 所示 (右)。接受域较大的特征具有广泛的泛化性，但似乎也会将学习函数限制为粗略的近似，无法使识别比接受域的宽度更精细。令人高兴的是，事实并非如此。从一个点到另一个点的初始泛化实际上是由接受域的大小和形状控制的，但是灵敏度，最终可能的最佳辨别能力，更多的是由特征的总数控制的。示例 9.3: 粗编码粗编码粗编码粗编码粗编码的粗编码粗编码对接收域大小学习的影响。利用基于粗编码的线性函数逼近 (9.7) 学习一维方波函数 (如图 9.8 所示)。这个函数的值被用作目标 Ut。只有一个维度，接受域是间隔而不是圆。学习是重复三种不同大小的间隔: 窄、中、宽，如图底部所示。这三种情况都具有相同的特征密度，大约是学习函数的 50 倍。在此范围内随机生成训练示例。步长参数 = 0.2 n , 其中 n 是一次出现的特性的数量。图 9.8 显示了在学习过程中在这三种情况下学习到的函数。注意，特征的宽度在早期学习中有很强的影响。具有广泛的特征，泛化趋于广泛; 由于特征较窄，只有每个训练点的近邻被改变，使得函数学习起来更加崎岖不平。然而，最终的功能只受特征宽度的影响。接受域形状对泛化影响较大，但对渐近解质量影响不大。

狭窄的特性媒介的功能广泛的特性

图 9.8: 特征宽度对初始泛化 (第一行) 的强影响和对渐近精度 (最后一行) 的弱影响的例子。

9.5.4 瓦片编码

块编码是一种用于多维连续空间的粗编码形式，具有灵活性和计算效率。它可能是现代顺序数字计算机最实用的特性表示。在块编码中，特征的接受域被分组到状态空间的分区中。每个这样的分区都称为瓦片，分区的每个元素都称为瓦片。例如，二维状态空间的最简单的 tiling 是一个均匀网格，如图 9.9 所示。这里的磁贴或接收域是正方形而不是图 9.6 中的圆。如果只使用这个单一的瓦片，那么白点表示的状态将由它所属的单个特性表示; 泛化对于同一块内的所有状态都是完整的，而对于它之外的状态则是不存在的。只有一个 tiling，我们不会有粗糙的编码，只是一个状态聚合的例子。

图 9.9: 有限的二维空间上的多个重叠网格倾斜。这些瓷砖在每个维度中，相互抵消的量是一致的。

为了获得粗编码的优势，需要重叠的接收域，根据定义，分区的块不会重叠。为了获得真正的粗代码块编码，使用多个倾斜，每一个偏移量都是块宽度的一小部分。图 9.9 右边显示了一个带有四个倾斜的简单案例。每一种状态，如白色点所示，在四种倾斜中每一种都正好落在一个瓦片上。这四个块对应于状态发生时变得活跃的四个特性。具体地说，特征向量 x(s) 对于每个瓦片中的每个瓦片都有一个组件。在本例中有 4×4×4 = 64 组件, 这一切都将成为 0 除了相对应的四个年代属于的瓷砖。图 9.10 显示了在 1000 状态随机漫步示例上，多个偏移倾斜 (粗编码) 比单个平铺的优点。块编码的一个直接的实际好处是，由于它与分区一起工作，所以一次活动的特性的总数对于任何状态都是相同的。每个瓦片上都有一个特性，所以显示的特性的总数总是与倾斜的数量相同。这使得步长参数 , 设置在一个简单、直观的方法。例如, 选择 = 1 n n 是这个数

图 9.10: 为什么使用粗编码。显示的是在 1000 状态随机上的学习曲线用单片多片梯度蒙特卡罗算法进行例子瓷砖。1000 个状态的空间被视为一个单一的连续维度，每 200 个状态的瓷砖覆盖。多重倾斜被 4 个状态抵消。步长参数设置, 初步学习速率两种情况是一样的, = 0.0001 单一 50 瓷砖瓷砖和 = 0.0001/50。倾斜的结果是精确的单次学习。如果示例年代?→v 是训练, 然后不管之前估计,v̂(年代,wt), 新的估计将 v̂(年代,wt + 1)= v . 通常希望改变比这更慢, 以允许泛化和随机变化的目标输出。例如, 一个可能选择 = 1 10 n, 在这种情况下, 估计训练状态将十分之一的目标在一个更新的方法, 和周边国家将更少, 瓷砖的数量成正比的共同点。使用二进制特征向量也获得了计算优势。由于每个分量都是 0 或 1，组成近似值函数 (9.8) 的加权和几乎是微不足道的。一个简单地计算 n 的指数，而不是执行 d 乘法和加法。d 主动特征，然后将权重向量的 n 个对应分量相加。泛化发生在除训练过的状态之外的其他状态，如果这些状态落在任何一个相同的块中，与相同的块的数量成比例。甚至如何抵消彼此的倾斜也会影响泛化。如果它们在每个维度上都被均匀地偏移，如图 9.9 所示，那么不同的状态可以以不同的方式进行定性归纳，如图 9.11 的上半部分所示。八个子图中的每一个都显示了从训练状态到附近点的泛化模式。在这个例子中，有 8 个倾斜，因此在一个瓦片中的 64 个分区，这是很明显的，但都是根据这 8 个模式中的一个。注意，在许多模式中，均匀偏移是如何在对角线上产生强大的效果的。如果倾斜是不对称的，可以避免这些伪影，如图的下半部分所示。这些较低的泛化模式更好，因为它们都以训练状态为中心，没有明显的不对称性。

可能的概括为统一抵消瓷砖

图 9.11: 为什么不对称偏移量是块编码的首选。显示了从训练状态到附近状态的泛化强度，用小黑色加号表示八个瓷砖。如果倾斜是均匀偏移 (上图)，则泛化中存在对角线伪影和大量的变化，而非对称偏移则泛化更为球形和均匀。

所有情况下的倾斜都被每一个维度上的瓦片宽度的一小部分所抵消。如果 w 表示瓦片宽度，n 表示倾斜次数，则 w n 是一项基本单位。在小广场 w n 在一边，所有的状态激活相同的磁砖，都是一样的。特征表示，和相同的近似值。如果一个状态被 w 移动 n 在任何笛卡尔的方向上，特征表示都由一个组件/块改变。均匀偏移的倾斜度由这个单位距离互相抵消。对于二维空间，我们说每个瓦片被位移矢量 (1,1) 抵消，这意味着它与之前的瓦片被 w 抵消 n 乘以这个向量。在这些方面, 图 9.11 下半部分所示的不对称偏移倾斜被 (1,3) 的位移矢量所

抵消。大量研究了不同位移矢量对地砖编码泛化的影响 (Parks and Militzer, 1991; 一个,1991 分; 一个, 米勒和公园,

1991; 米勒，An, Glanz 和 Carter, 1990)，评估他们的同质性和对角工件的趋势，就像 (1,1) 位移向量所见。基于这项工作，Miller 和 Glanz(1996) 建议使用由第一个奇整数组成的位移向量。特别地，对于维 k 的连续空间，一个好的选择是使用第一个奇数 (1,3,5,7)。,2 k−1),n(瓷砖) 的数量设置为 2 的整数次幂大于或等于 4 k。这是我们做过的瓷砖生产图 9.11 的下半部,k = 2,n = 23 4 k 和位移矢量 (1,3)。在三维情况下, 前四个瓷砖将抵消总从基地位置 (0,0,0),(1、3、5),(2、6、10),(3、9、15)。开源软件可以有效地为任何 k 做这样的倾斜是容易得到的。在选择瓦片策略时，必须选择倾斜的数量和瓦片的形状。倾斜的数量，以及贴图的大小，决定了渐近逼近的分辨率或细度，如一般的粗编码所示，如图 9.8 所示。瓦片的形状将决定泛化的性质，如图 9.7 所示。在图 9.11(下图 9.11) 所示的每个维度中，方砖的泛化程度大致相同。在一个维度上延伸的瓷砖，如图 9.12(中) 中的条纹倾斜，将沿着这个维度推广推广。图 9.12(中间) 中的倾斜在左边也更密集和更薄，在水平方向上沿该维度的较低值促进了识别。图 9.12(右) 中的对角线条纹条将促进沿一个对角线的泛化。在更高的维度中，轴向条纹对应于忽略一些倾斜的维度，也就是超平面切片。图 9.12(左) 所示的不规则倾斜也是可能的，尽管在实践中很少出现，而且超出了标准软件。

不规则的 b) Log 条纹 c) 斜条纹。

图 9.12: 倾斜不需要网格。它们可以是任意形状和非均匀的在很多情况下计算效率很高。

在实践中，通常需要在不同的倾斜中使用不同形状的瓷砖。例如，可以使用一些垂直条纹倾斜和一些水平条纹倾斜。这将鼓励在这两个维度上进行泛化。然而，仅凭条纹倾斜就不可能了解水平坐标和垂直坐标的特定结合具有独特的价值 (无论学习到什么，它都会以相同的水平坐标和垂直坐标为基础进入状态)。为此，需要如图 9.9 所示的连接矩形块。有了多重顶点——一些是水平的，一些是垂直的，还有一些是结合点——我们就可以得到所有的东西: 偏好沿着每个维度进行归纳，同时也有能力学习连接的特定值 (参见 Sutton，

1996 为例)。倾斜的选择决定了一般化，并且在这个选择能够被有效地自动化之前，重要的是，tile 编码能够灵活地做出选择，并且以一种对人们有意义的方式。另一个减少内存需求的有用的技巧是 hash- a，它是一种连续的伪随机崩溃，将一个大的 tiling 折叠成一个小得多的块。散列产生的块由非连续的、不相交的区域组成，这些区域在整个州内随机分布

一个瓷砖空间，但那仍然构成一个详尽的分区。例如，一个块可能由右边显示的四个子块组成。通过散列，内存需求通常会被较大的因素所减少，而性能损失很小。这是可能的，因为只有一小部分的状态空间需要高分辨率。哈希将我们从维数的诅咒中解放出来，因为内存需求不必在维数上呈指数级增长，而只需满足任务的实际需求。块编码的开源实现通常包括高效散列。练习 9.4 假设我们认为两个状态维度中的一个比另一个更有可能对值函数产生影响，那么泛化应该主要是在这个维度上而不是沿着这个维度。什么样的倾斜可以使用这种先验知识的优势是什么？？

9.5.5 径向基函数

径向基函数 (RBFs) 是粗编码对连续值特征的自然泛化。而不是每一个特征都是 0 或 1，它可以是区间 [0,1] 中的任何东西，反映了特征存在的不同程度。典型的 RBF 特性, 习近平, 高斯 (钟形) 响应 xi(s) 只依赖状态, 之间的距离, 和功能的原型或中心状态,ci, 和相对于功能的宽度, i:

习近平 (年代)。= 经验

− | | 年代 −ci | | 2 2 2 我

.

标准或距离度量当然可以选择任何看起来最适合当前状态和任务的方式。下图显示了一个具有欧几里得距离度量的一维示例。

图 9.13: 一维径向基函数。

相对于二进制特征，RBFs 的主要优点是它们产生的近似函数变化平滑且可微。尽管这很吸引人，但在大多数情况下，它并没有什么实际意义。然而，广泛的研究已经在 tile coding (An, 1991) 的语境中对 RBFs 等分级响应函数进行了广泛的研究。米勒 et al .,1991;et al .,1991; 莱恩，汉德尔曼和盖尔芬德，1992)。所有这些方法都需要大量额外的计算复杂性 (通过块编码)，并且当有两个以上的状态维度时，常常会降低性能。在高维情况下，瓷砖的边缘要重要得多，而且已经证明很难在边缘附近得到控制良好的渐变瓷砖激活。RBF 网络是一种基于 RBFs 的线性函数逼近器。学习是由方程 (9.7) 和 (9.8) 定义的，与其他线性函数逼近器完全一样。此外，RBF 网络的一些学习方法也会改变特征的中心和宽度，使它们进入非线性函数逼近器的领域。非线性方法可以更精确地拟合目标函数。RBF 网络，尤其是非线性 RBF 网络的缺点是计算复杂度更高，而且通常在学习之前需要更多的手动调优，这是健壮且高效的。

9.6 手动选择步长参数

大多数 SGD 方法要求设计师选择合适的步长参数 。理想情况下，这个选择是自动的，在某些情况下是自动的，但是在大多数情况下，手动设置它仍然是常见的做法。要做到这一点，并更好地理解算法，开发一些关于步长参数的作用的直觉感觉是很有用的。我们能概括地说它应该如何设置吗? 不幸的是，理论上的考虑没有多大帮助。随机逼近理论给出了一个缓慢减小的步长序列的条件 (2.7)，它足以保证收敛，但这些条件往往导致学习速度太慢。经典选择 t = 1 / t, 生产样品平均表格 MC 方法, 不适合 TD 方法, 对于不稳定问题, 或任何方法使用函数近似。对于线性方法，有设置最优矩阵步长的递归最小二乘方法，这些方法可以扩展到时间差异学习，如第 9.8 节描述的 LSTD 方法，但是这些方法需要 O(d2) 步长参数，或者是 d 乘以我们正在学习的参数。由于这个原因，我们把它们排除在最需要函数逼近的大问题中。为了直观地了解如何手动设置步长参数，最好是暂时回到列表框中。我们可以理解, = 1 的步长将导致一个完整的样本误差的消除后一个目标 (见 (2.4) 的步长)。正如第 201 页所讨论的，我们通常希望学习得慢一些。在表格的情况下, = 1 的步长 10 大约需要 10 经验收敛他们的目标, 在 100 年, 如果我们想学习经验我们会使用 = 1 100。一般来说, 如果 = 1 ，则状态的列表估计值将方法的意思是目标, 最近的目标有最大的影响, 的经历后的状态。

对于一般的函数逼近，对于一个状态的经历的数量没有如此清晰的概念，因为每个状态可能在不同程度上与所有其他状态相似或不同。然而，在线性函数近似的情况下，有一个相似的规则给出相似的行为。假设你想学在 经验实质上相同的特征向量。然后给出了线性 SGD 方法步长参数设置的一个很好的经验法则

。 =

E

x ?

−1 , (9.19)

其中 x 是一个随机特征向量，从与输入向量相同的分布中选择，在 SGD 中。当特征向量的长度变化不大时，该方法最有效; 理想 x ?x 是一个常数。练习 9.5 假设您使用的是瓷砖编码一个 7 维连续状态空间转换成二进制特征向量估计状态值函数 v̂(s,w) v (年代)。你相信尺寸不相互作用强烈, 所以你决定使用八瓷砖的每个维度分别 (条纹瓷砖),7×8 = 56 瓷砖。此外，如果在维度之间有一些成对的交互，您也需要全部。7 2 ?

= 21 尺寸对和瓷砖每一对结合与矩形瓷砖。你让两个瓷砖每一对维度, 进行总计 21×2 + 56 = 98 瓷砖。考虑到这些特征向量，你怀疑你仍然需要平均出一些噪音，所以你决定你想要学会循序渐进，在学习接近它的渐近线之前，用同样的特征向量做大约 10 个演示。什么步长参数 你会使用吗? 为什么? ?

9.7 非线性函数逼近: 人工神经网络。

人工神经网络被广泛应用于非线性函数逼近。神经网络是由相互联系的单元组成的网络，

它们具有神经元的某些特性，神经元是神经系统的主要组成部分。ANNs 有着悠久的历史，在训练深度分层 ANNs(深度学习) 方面的最新进展是机器学习系统最令人印象深刻的能力，包括强化学习系统。在第 16 章中，我们描述了几个使用 ANN 函数逼近的强化学习系统的令人印象深刻的例子。图 9.14 显示了一个通用的前馈神经网络，这意味着网络中没有循环，也就是说，网络中没有路径可以让一个单元的输出影响它的输入。图中的网络有一个输出层，由两个输出单元组成，一个输入层有四个输入单元，以及两个“隐藏层”: 既不是输入层也不是输出层的层。实值权重与每个链接相关联。重量大致相当于真实神经网络中突触连接的效力 (见第 15.1 节)。如果一个 ANN 在它的连接中至少有一个循环，那么它是一个循环，而不是前馈的 ANN。虽然前馈和周期性 ann 都被用于强化学习，但这里我们只看比较简单的前馈情况。

图 9.14: 一个具有四个输入单元、两个输出单元和两个隐藏层的通用前馈神经网络。

单元 (图 9.14 中的圆圈) 通常是半线性单元，这意味着它们计算输入信号的加权和，然后将一个非线性函数 (称为激活函数) 应用于结果，以产生单元的输出或激活。使用不同的激活函数, 但他们通常 s 形, 或乙状结肠, 物流等功能函数 f(x)= 1 /(1 + e−x), 尽管有时整流器非线性 f(x)= max(0,x)。一个阶跃函数 f(x)= 1 如果 x , 和 0 否则, 结果在一个二进制单元阈值 。网络输入层中的单元在将它们的激活设置为外部提供的值时有所不同，这些值是网络正在逼近的函数的输入。前馈神经网络的每个输出单元的激活是网络输入单元上激活模式的非线性函数。函数由网络的连接权重参数化。没有隐藏层的 ANN 只能表示可能的输入输出函数的很小一部分。然而，具有一个包含足够大的有限数量的乙状结肠单元的单一隐藏层的 ANN，可以在网络输入空间的紧凑区域上以任何程度的精确度近似任何连续函数 (Cybenko, 1989)。这也适用于其他非线性激活函数满足条件温和, 但是非线性是至关重要的: 如果所有的单位在一个多层前馈神经网络有线性的激活函数, 整个网络就相当于一个网络没有隐藏层 (因为线性函数的线性函数是线性)。尽管这“万能逼近”属性 one-hidden-layer ann, 经验和理论表明, 近似许多人工智能任务所需的复杂的功能是由 easier-indeed 可能 require-abstractions 层次组成的许多低级别的抽象层, 即抽象由深架构如人工神经网络有许多隐藏层。(详见 2009 年的 Bengio 详细回顾。) 深 ANN 的连续层逐渐地计算

网络的“原始”输入的抽象表示，每个单元提供一个特性，用于对网络的总体输入-输出函数进行分层表示。因此，训练 ANN 的隐藏层是一种自动创建适合于给定问题的特性的方法，这样就可以在不依赖手工制作的特性的情况下生成分层表示。这对人工智能来说是一个持久的挑战，也解释了为什么具有隐藏层的人工智能学习算法近年来受到如此多的关注。ANNs 通常通过随机梯度法学习 (第 9.3 节)。每一权值的调整方向都是为了提高网络的整体性能，目标函数是最小化或最大化。在最常见的监督学习案例中，目标函数是预期的错误或损失，而不是一组标记的训练示例。在强化学习中，ANNs 可以使用 TD 错误来学习价值函数，或者可以像梯度班迪特 (第 2.8 节) 或政策梯度算法 (第 13 章) 那样，以最大化预期的回报为目标。在所有这些情况下，需要估计每个连接权重的变化将如何影响网络的整体性能，换句话说，考虑到所有网络权重的当前值，估计目标函数对每个权重的偏导。梯度是这些偏导数的向量。对于具有隐藏层的人工神经网络 (如果这些单元具有可微的激活功能)，最成功的方法是反向传播算法，它由交替向前和向后穿过网络组成。每个前向传递计算每个单元的激活，给定网络输入单元的当前激活。每次向前传球后，向后传球有效地计算每一重量的偏导数。(和其他随机梯度学习算法一样，这些偏导数的向量是真实梯度的估计值。) 在第 15.10 节中，我们讨论了使用增强学习原理而不是反向传播的隐藏层进行训练的方法。这些方法不如反向传播算法有效，但它们可能更接近于真实神经网络的学习方式。对于具有 1 或 2 个隐藏层的浅层网络，反向传播算法可以产生良好的结果，但对于较深的 ANNs，它可能不太适用。事实上，训练一个有 k + 1 隐藏层的网络实际上比训练一个有 k 隐藏层的网络性能差，即使深层网络可以代表浅层网络所能代表的所有功能 (Bengio, 2009)。解释这样的结果并不容易，但有几个因素很重要。首先，典型的深度神经网络中权重的大量存在，使其难以避免过度拟合的问题，即不能正确地推广到尚未对网络进行训练的情况。第二, 反向传播不适合深 ann 因为偏导数计算的向后传球要么衰变迅速向网络的输入端, 使学习的深度层极慢, 或偏导数快速增长对网络的输入端, 使学习不稳定。处理这些问题的方法在很大程度上要归功于使用深的 ANNs 系

统所取得的许多令人印象深刻的最新成果。超拟合是任何函数逼近方法调整函数的问题

在有限的训练数据的基础上有许多自由度。不依赖有限的训练集的在线强化学习问题较少，但有效的归纳仍然是一个重要的问题。一般来说，过拟合是 ANNs 的一个问题，但对深的 ANNs 来说尤其如此，因为它们往往有大量的重量。人们已经开发出许多减少过拟合的方法。其中包括当验证数据 (交叉验证) 的性能开始下降时停止训练 (交叉验证)，修改目标函数以阻止逼近的复杂性 (正则化)，引入权重之间的依赖关系以减少自由度 (例如，权重共享)。Srivastava、Hinton、Krizhevsky、Sutskever 和 Salakhutdinov(2014) 等人提出的辍学率法是一种特别有效的深部 ANNs 拟合方法。在训练过程中，单位与他们的连接一起被随机地从网络中删除 (退出)。这可以被认为是训练了大量的“瘦”网络。结合测试时这些稀疏网络的结果，可以提高泛化性能。辍学率法通过将一个单元的每个出站重量乘以该单元在训练中被保留的概率来有效地近似这个组合。Srivastava 等发现，该方法显著提高了泛化性能。它鼓励单个隐藏单元学习与其他特性的随机集合一起工作的特性。这增加了隐藏单元所形成的特征的多样性，这样网络就不会过度专门化到罕见的情况。Hinton、Osindero 和 Teh(2006) 在他们的研究中，利用深度信念网络，即与这里讨论的深度 ANNs 密切相关的分层网络，向解决训练深层 ANN 的问题迈出了重要的一步。在他们的方法中，最深层的层次一次一个的训练使用一个无监督的学习算法。无监督学习不依赖于整体目标函数，可以提取出捕获输入流统计规律的特征。最深的层首先被训练，然后由这个训练过的层提供输入，下一个最深的层被训练，等等，直到网络的所有或许多层的权重被设置为现在作为监督学习的初始值。然后通过反向传播对整个目标函数进行微调。研究表明，这种方法通常比用随机值初始化权值的反向传播效果更好。通过这样的方式来训练的网络性能越好，这可能是由许多因素造成的，但有一种观点认为，这种方法将网络置于一个权重空间的区域，而基于梯度的算法能够取得良好的进展。批标准化 (Ioffe 和 Szegedy, 2015) 是另一种更容易训练深度 ANNs 的技术。人们早就知道，如果网络输入是规范化的，比如将每个输入变量调整为零均值和单位方差，那么学习 ANN 会更容易。深度 ANNs 训练的批处理规范化在深层输入到下一层之前将其输出规范化。Ioffe 和 Szegedy(2015) 利用训练样本的子集或“小批量”的统计数据，将这些层间信号规范化，以提高深部 ANNs 的学习速度。另一种有助于训练深层记忆的技术是深度剩余学习 (他、张、任、孙，2016)。有时，我们更容易了解一个函数与另一个函数的区别

恒等函数比学习函数本身更重要。然后，将这个差值或剩余函数添加到输入中，就会产生所需的函数。在深度 ANNs 中，只需在块周围添加快捷键或跳过连接，就可以创建一组层来学习一个剩余函数。这些连接将输入添加到块的输出中，不需要额外的权重。他等人 (2016) 对这种方法进行了评估，使用了深度卷积网络，在每对相邻层之间都有跳过连接，发现在基准图像分类任务中没有跳过连接的网络有了实质性的改进。在第 16 章中所描述的“Go”游戏的强化学习应用中使用了批量标准化和深度剩余学习。深层神经网络被证明在应用中非常成功，包括令人印象深刻的强化学习应用 (第 16 章)，它就是深层卷积网络。这种类型的网络专门用于处理空间数组 (如图像) 中的高维数据。它的灵感来自于早期视觉处理在大脑中的工作原理 (LeCun, Bottou, Bengio 和 Haffner, 1998)。由于其特殊的结构，深层卷积网络可以通过反向传播进行训练，而不需要使用上面描述的方法来训练深层层。图 9.15 展示了深层卷积网络的架构。这个例子，来自 LeCun 等人 (1998)，旨在识别手写字符。它由交替的卷积层和次采样层组成，然后是几个完全连接的最后层。每个卷积层产生许多特征映射。特征映射是一个活动的模式在一个数组的单元, 在每个单元上执行相同的操作数据的接受域, 数据的一部分它“看到”的前层 (或从外部输入的第一个卷积层)。特征图的单位是相同的，除了它们的接收域都是相同的大小和形状，被移动到输入数据数组的不同位置。同一地形图中的单元拥有相同的权重。这意味着，无论功能映射位于输入数组的哪个位置，它都会检测到相同的功能。在网络在图 9.15 中, 例如, 第一次卷积层产生的 6 个特征图, 每个 28×28 个单位组成。每个单元在每个特性图 5×5 接受域, 这些接受领域重叠四列 (在本例中,4

图 9.15: 深度卷积网络。经批准，重新出版 IEEE，从基于梯度的学习应用到文档识别，LeCun, Bottou, Bengio 和 Haffner，第 86 卷，1998; 许可通过版权许可中心，Inc。

行)。因此，6 个特征图中的每一个都由 25 个可调权值指定。深度卷积网络的子采样层降

低了空间分辨率特征图。在一个子采样层中，每一个特征图都由在前卷积层的特征图中接受单元域上的平均单元组成。例如, 每个单元的 6 个特征图前的二次抽样层网络图 9.15 平均在 2×2 重叠接受域在产生的特征图的第一个卷积层, 导致六 14×14 个特征图。子采样层降低了网络对检测到的特征的空间位置的敏感性，也就是说，它们有助于使网络的响应在空间上不变性。这是很有用的，因为在图像中某个位置检测到的特性可能在其他位置也有用。在 annis 的设计和培训方面的进步——我们只提到了少数——都有助于强化学习。虽然目前的强化学习理论大多局限于使用表格函数或线性函数逼近方法的方法，但值得注意的强化学习应用的显著性能很大程度上归功于多层神经网络的非线性函数逼近。我们将在第 16 章中讨论其中的一些应用。

9.7 最小二乘道明

到目前为止，我们在本章讨论过的所有方法都需要计算与参数数量成正比的时间步长。然而，有了更多的计算，就可以做得更好。在本节中，我们将介绍一种线性函数逼近的方法，它可以被认为是这种情况下所能做的最好的方法。正如我们在 9.4 TD(0) 中建立的线性函数逼近渐近收敛 (对于适当减小的步长) 到 TD 不动点:

西医 =−1 b,

一个。= E

xt(xt− xt + 1)? 和 b。= E(Rt + 1 xt)。有人可能会问，为什么我们必须迭代地计算这个解? 这是浪费数据! 通过计算 A 和 b 的估计值，然后直接计算 TD 的不动点，难道不能做得更好吗? 最小二乘 TD 算法 (通常称为 LSTD) 就是这样做的。它形成了自然的估计

吗? 在。= t−1 ?k = 0 xk(xk− xk + 1)?+ I 吗? 英国电信。= t−1 ?k = 0 Rt + 1 xk,(9.20)

我是单位矩阵, I, 对于一些小的 > 0, 确保吗? 在都是可逆的。看起来这些估计都应该除以 t，事实上它们应该; 根据定义，这些是 t 乘以 A 和 t 乘以 b 的估计值。

然而，当 LSTD 使用这些估计来估计 TD 的不动点时，额外的 t 因子就会抵消掉 wt 。= 吗?−1 t ? 英国电信。(9.21)

该算法是线性 TD(0) 中数据效率最高的一种算法，但计算开销也较大。回想一下，半梯度 TD(0) 只需要 O(d) 的内存和每步计算。LSTD 有多复杂? 正如上面所写的那样，复杂性似乎随着 t 而增加，但是 (9.20) 中的两个近似可以用我们之前讨论过的技术 (例如，在第 2 章中) 增量地实现，这样它们就可以在每一步的固定时间内完成。即便如此，更新的目的是什么? 将涉及一个外积 (一个列向量乘以一个行向量)，因此将是一个矩阵更新; 它的计算复杂度是 O(d2)，当然，它所需要的内存是多少? 矩阵是 O(d2) 一个潜在的更大的问题是我们的最终计算 (9.21) 使用的是? 一般逆向计算的计算复杂度为 O(d3)。幸运的是，我们的特殊形式的一个矩阵的逆矩阵——外部产物的和——也可以只通过 O(d2) 计算来增量地更新

对于 t > 0，有?A0 。= I。虽然身份 (9.22)，被称为 Sherman-Morrison 公式，表面上是复杂的，它只涉及向量矩阵和向量向量乘法，因此只有 O(d2)。这样我们就可以存储逆矩阵了?−1 t, 维持它与 (9.22), 然后使用 (9.21), 所有只有 O(d2) 内存和每一步计算。完整的算法在下一页的框中给出。当然，O(d2) 仍然比半梯度 TD 的 O(d) 昂贵得多。LSTD 的更大数据效率是否值得这个计算费用取决于 d 的大小，快速学习的重要性，以及系统其他部分的开销。LSTD 不需要步长参数的事实有时也被吹捧，但这一优点可能被夸大了。LSTD 并不需要一个步长, 但它确实需要 ; 如果 选择太小的逆顺序可以大相径庭, 如果选择 太大学习是放缓。此外，LSTD 缺少一个步长参数意味着它永远不会忘记。这有时是可取的, 但它是有问题的, 如果目标政策 与强化学习和谷歌价格指数的变化。在控制应用程序中，LSTD 通常必须与其他一些机制相结合，以诱导 forgeting，提出不需要步长参数的任何初始优势。

9.8 基于内存的函数近似

到目前为止，我们已经讨论了逼近值函数的参数化方法。在这种方法中，学习算法调整函数形式的参数，以便在问题的整个状态空间中估计值函数。每次更新, 年代?→g, 使用的是一个训练的例子学习算法改变参数, 目的是减少近似误差。在更新之后，可以丢弃训练示例 (尽管它可能被保存以再次使用)。当需要一个状态的近似值 (我们称之为查询状态) 时，只需使用学习算法生成的最新参数在该状态下对函数进行评估。基于内存的函数逼近方法是非常不同的。它们只是在到达时将训练示例保存在内存中 (或者至少保存示例的子集)，而不更新任何参数。然后，每当需要查询状态的值估计时，就从内存中检索一组示例并用于计算查询状态的值估计。这种方法有时被称为延迟学习，因为处理训练示例要延迟到查询系统以提供输出时。基于记忆的函数近似方法是非参数方法的主要例子。与参数化方法不同，逼近函数的形式并不局限于一个固定的参数化函数类，如线性函数或多项式，而是由训练示例本身决定，以及将它们与查询状态的输出估计值相结合的一些方法。随着越来越多的训练例子在内存中积累，我们期望非参数方法能够对任何目标函数产生越来越精确的逼近。

根据如何选择存储的训练示例以及如何使用它们响应查询，有许多不同的基于内存的方法。在这里，我们将重点放在局部学习方法上，这些方法仅在当前查询状态的附近局部近似一个值函数。这些方法检索一组训练的例子从记忆的状态被认为是最相关的查询状态, 在相关性通常取决于国家之间的距离: 越接近训练实例的状态查询状态, 更相关的是, 距离可以以许多不同的方式定义。在给查询状态一个值之后，将丢弃局部逼近。基于内存的方法最简单的例子是最近的邻居方法，它只在内存中找到最接近查询状态的实例，并返回该实例的值作为查询状态的近似值。换句话说，如果查询状态是 s 和 s??→g 内存中的示例在哪个年代? 是最近的状态, 然后返回 g s。稍微复杂的近似值是加权平均的方法检索一组最近邻的例子并返回一个目标值的加权平均, 权重的一般减少与增加查询状态和状态之间的距离。局部加权回归类似，但它通过参数逼近方法将加权误差度量 (如 (9.1) 降至最小，使权重依赖于距离查询状态的距离，使其与一组最接近状态的值吻合。返回的值是对查询状态下的局部拟合曲面的求值，然后将局部逼近曲面丢弃。由于是非参数的，基于内存的方法比不将逼近限制在预先指定的函数形式的参数方法具有优势。这允许随着数据的积累而提高准确性。基于记忆的局部逼近方法有其他的性质，使它们非常适合于强化学习。由于轨迹采样在增强学习中非常重要，如第 8.6 节所讨论的，基于内存的局部方法可以将函数逼近集中在真实轨迹或模拟轨迹中访问的状态 (或状态-动作对) 的局部邻域上。可能不需要全局近似，因为状态空间的许多区域永远 (或几乎永远不会) 到达。此外，基于内存的方法允许代理的经验对当前状态附近的值估计产生相对直接的影响，与参数方法需要增量地调整全局逼近的参数形成对比。避免全局逼近也是解决维度诅咒的一种方法。例如, 对于与 k 维状态空间, 一个表存储全局近似方法需要记忆指数 k。另一方面, 在存储的例子基于内存的方法, 每个例子都需要内存 k 成正比, 和所需的内存来存储, 说,n 是线性的例子在 n。没有什么是指数 k 或 n。当然, 剩下的关键问题是一个基于内存的方法是否能回答查询很快是有用的一个代理。一个相关的问题是，随着内存的增加，速度会如何降低。在大型数据库中查找最近的邻居可能会花费太长时间，在许多应用程序中都不太实用。

基于内存的方法的支持者已经开发出加速最近邻居搜索的方法。使用并行计算机或专用硬件是一种方法; 另一种方法是使用特殊的多维数据结构来存储训练数据。这个应用程序研究的一个数据结构是 k-d 树 (k 维树的缩写)，它递归地将 k 维空间分割成一棵二叉树的节点。根据数据量和数据在状态空间上的分布方式，使用 k-d 树的最近邻搜索可以快速地消除搜索邻域中的大片区域，使搜索在一些朴素搜索时间过长的问题中可行。局部加权回归还需要快速的方法来进行局部回归计算，这些计算必须重复来回答每个查询。研究人员已经开发了许多方法来解决这些问题，包括忘记条目的方法，以便使数据库的大小保持在一定范围内。本章末尾的书目和历史评论部分指出了一些相关的文献，包括描述基于记忆的学习在强化学习中的应用的论文选集。

9.9 基于函数逼近

基于内存的方法，如上面描述的加权平均和局部加权回归方法，依赖于给示例赋权??→g 数据库中根据年代之间的距离? 查询状态为 s。分配这些权重的函数称为核函数，或者简单地称为核函数。加权平均和局部加权回归方法, 例如, 一个内核函数 k:R→ 分配权重之间的距离。更一般地说，重量不必依赖于距离; 它们可以依赖于国家间相似度的其他度量。在这种情况下, 凯西:S×S→R, 这 k(S,S) 给出的重量数据年代呢? 它对回答关于 s 的问题的影响。从稍微不同的角度来看，k(s, s?) 是对从 s 到 s 的泛化强度的度量。对于 s。核函数的数值表示任何状态的相关知识与其他状态的关系。例如，图 9.11 所示的块编码泛化的强度对应于由于均匀和不对称的块偏移而产生的不同的核函数。虽然块编码在其操作中没有显式地使用内核函数，但它是根据内核函数进行推广的。事实上，正如我们在下面讨论的那样，由线性参数函数逼近所产生的泛化强度总是可以用核函数来描述的。内核回归是一种基于内存的方法，它计算存储在内存中的所有示例的内核加权平均，并将结果分配给查询状态。如果 D 是存储的示例集，而 g(s?) 表示状态 s 的目标? 在存储的示例中，然后内核回归逼近目标函数，在本例中，值函数依赖于 D，如

v̂(s、D)= ? D s ? k(s,s)g(s ?)。(9.23)

上面描述的加权平均方法是一种特殊情况，只有当 s 和 s 时 k(s, s?) 才是非零的。它们之间的距离很近，所以不需要对整个 D 进行求和。常见的核是用于 RBF 函数近似的高斯径向基函数 (RBF)，如第 9.5.5 节所述。在这里描述的方法中，RBFs 是中心和宽度从一开始就固定的特性，中心可能集中在许多示例预计会下降的区域，或者在学习过程中以某种方式进行调整。这是一种线性参数化方法，其参数为每个 RBF 的权重，通常通过随机梯度或半梯度下降来学习。近似的形式是预先确定的 RBFs 的线性组合。使用 RBF 内核的内核回归在两个方面与此不同。首先，它是基于内存的:RBFs 以存储示例的状态为中心。第二，它是非参数化的: 没有参数可以学习; 查询的响应由 (9.23) 给出。当然，在实际实现内核回归时需要解决许多问题，这些问题超出了我们简短讨论的范围。然而，事实证明，任何线性参数回归方法，如我们在第 9.4 节中描述的，状态由特征向量 x(s) = (x1(s)，x2(s)，⋯,xd(s))?k(s, s?) 是 s 和 s 的特征向量表示的内积; 这是

k(s,s)= x(s)? x(?)。(9.24)

使用这个核函数的核回归产生的近似与线性参数方法使用这些特征向量并学习相同的训练数据时所产生的近似相同。我们跳过了任何现代机器学习文本 (如 Bishop(2006)) 中都可以找到的数学理由，只简单地指出一个重要的含义。与其构造线性参数函数近似器的特征，还不如直接构造核函数而不涉及特征向量。并不是所有的核函数都可以表示为特征向量的内积 (9.24)，但是可以像这样表示的核函数可以在等价的参数化方法上提供显著的优势。对于许多特征向量集，(9.24) 有一个紧凑的函数形式，可以在没有任何计算的 d 维特征空间中进行计算。在这些情况下，与直接使用由这些特征向量表示的状态的线性参数方法相比，核回归要简单得多。这就是所谓的“内核技巧”，它允许在扩展特性空间的高维空间中有效地工作，而实际上只使用存储的训练示例集。核心技巧是许多机器学习方法的基础，研究人员已经展示了它有时是如何有益于强化学习的。

9.10 对政策学习的深入研究: 兴趣和重点

到目前为止，我们在本章中所考虑的算法已经把遇到的所有状态都平等地对待，就好像它们都同样重要一样。然而，在某些情况下，我们比其他国家更感兴趣。例如，在折现的偶发问题中，我们可能更感兴趣的是准确地估计事件中早期状态的价值，而不是在后来的状态中，打折可能会使奖励对起始状态的价值不那么重要。或者，如果正在学习一个行为价值函数，那么精确地评估价值远低于贪婪行为的糟糕行为可能就不那么重要了。函数逼近资源总是有限的，如果以更有针对性的方式使用它们，那么性能就可以得到改善。我们平等对待所有国家的

一个原因是，我们正在根据政策的分配情况进行更新，因为有更强的理论结果可以用于半梯度法。回想一下，on-policy 分布定义为在执行目标策略时在 MDP 中遇到的状态分布。现在我们将显著地推广这个概念。我们不会为 MDP 提供一个政策上的分配，我们会有很多。它们都有一个共同点，它们都是在执行目标政策时，在轨迹中遇到的状态的分布，但它们在轨迹如何，在某种意义上，初始化上是不同的。我们现在介绍一些新概念。首先我们引入一个非负标量测度，一个叫做兴趣的随机变量，表示我们对 t 时刻的状态 (或状态-动作对) 进行精确估值的程度。如果我们完全关心它，它可能是 1，尽管它被正式地允许取任何非负值。兴趣可以随意设定; 例如, 它可能取决于时间 t 的轨迹或学习参数在时间 t。VE 中的分布 (9.1) 被定义为状态的分布时遇到的目标政策后, 加权的兴趣。其次，我们引入另一个非负标量随机变量，强调 Mt。这个标量乘以学习更新，从而强调或不强调在 t 时刻完成的学习

wt + n 。= t + n−1 + Mt(Gt:t + n−v̂(圣、wt + n−1)] v̂(St,wt + n−1),0 t < t(9.25)

由 (9.16) 给出的 n 阶回报，以及由利息递归确定的重点:

太 =,+ nMt−n,0 t < t, (9.26)

与太。= 0，对于所有 t < 0。这些方程包括蒙特卡罗的情况，的 Gt:t + n = Gt, 所有的更新一集结束,n = t−t, 太 =。示例 9.4 说明了兴趣和重点是如何导致更精确的价值估计的。

9.11。更深入地研究政策学习: 兴趣和重点。235 年

9.11 总结

强化学习系统必须能够概括，如果它们适用于人工智能或大型工程应用。为了实现这一点，任何现有的用于监督学习函数近似的方法都可以简单地使用—将每个更新作为一个训练示例。也许最合适的监督学习方法是使用参数化函数逼近的方法，其中策略是由权向量 w 参数化的。虽然权向量有很多分量，但状态空间更大，我们必须满足于一个近似解。我们定义了均方值错误,VE(w), 作为一个测量的误差值 v w(s) 的权向量 w 在政策下分布, 。VE 为我们提供了一种清晰的方法来对不同的价值函数逼近进行排序。为了找到一个好的权值向量，最常用的方法是随机梯度下降法。在本章中，我们重点讨论了一个固定政策的政策案例，也就是政策评估或预测; 这种情况下的自然学习算法是 n-step semi-gradient TD, 包括梯度蒙特卡罗和 semi-gradient TD(0) 算法作为特殊情况分别当 n =∞ 和 n = 1。半梯度 TD 方法不是真正的梯度方法。在这种 bootstrapping 方法 (包括 DP) 中，权重向量出现在更新目标中，但在计算梯度时没有考虑到这一点——因此它们是半梯度方法。因此，他们不能依赖经典的 SGD 结果。然而，在线性函数逼近的特殊情况下，半梯度方法可以得到很好的结果，其中值估计是特征的和乘以相应的权值。线性情况在理论上是最容易理解的，当提供适当的特征时，在实践中也很有效。选择特征是在强化学习系统中增加先验知识的最重要的方法之一。它们可以被选择为多项式，但这种情况在强化学习中通常考虑的在线学习设置中推广得很差。更好的方法是根据傅立叶基，或者根据具有稀疏重叠接受域的粗编码形式来选择特征。块编码是一种粗编码的形式，它在计算效率和灵活性方面尤为突出。径向基函数对于一个或两个二维的任务是有用的，其中一个平滑变化的响应是重要的。LSTD 是最具数据效率的线性 TD 预测方法，但它要求计算与权值的平方成正比，而其他所有方法的权值都是线性的。非线性方法包括反向传播训练的人工神经网络和 SGD 的变异; 这些方法近年来在“深度强化学习”这个名字下变得非常流行。线性半梯度 n 阶 TD 保证在标准条件下收敛，对于所有 n，收敛到一个在最优误差范围内的 VE(蒙特卡罗方法渐近实现)。这个绑定总是严格高 n 和趋于 0 n→∞。然而, 在实践中非常高的 n 导致非常缓慢的学习, 和某种程度的引导 (n <∞) 通常是 preferrable, 正如我们看到在第七章表格 n-step 方法的比较和对比的表格 TD 和蒙特卡罗方法在第六章。

9.12 书目的和历史的言论

概化和函数逼近一直是强化学习的重要组成部分。Bertsekas 和 tsiklis(1996)、Bertsekas(2012) 和 Sugiyama 等人 (2013) 提出了函数逼近在强化学习中的最新进展。本文最后讨论了函数逼近在强化学习中的一些早期工作。

9.3 监督下最小化均方误差的梯度下降法学习是众所周知的。Widrow 和 Hoff(1960) 引入了最小均方 (LMS) 算法，这是一种典型的增量梯度下降算法。这个和相关算法的细节在许多文本中都有提供 (例如，Widrow 和 Stearns, 1985; 主教,1995; 杜达和 Hart,1973)。Semi-gradient TD(0) 首次探索萨顿 (1984、1988), 作为线性 TD( ) 算法的一部分, 我们将在第 12 章。描述这些自举方法的术语“半梯度”是本书第二版的新术语。在强化学习中最早使用状态聚合的可能是 Michie 和 Chambers 的 box 系统 (1968)。国家在强化学习中的聚集理论是由 Singh、Jaakkola 和 Jordan(1995)、Tsitsiklis 和 Van Roy(1996) 提出的。状态聚合从早期就被用于动态编程 (例如，Bellman, 1957 年 a)。

9.4 Sutton(1988) 证明了线性 TD(0) 与最小值的收敛性。已经解决的案例特征向量, x(s) 的模式:s 年代, 是线性无关的。与概率 1 的收敛性几乎同时被几个研究者证明 (Peng, 1993; 达扬 Sejnowski,1994;Tsitsiklis,1994;Gurvits, Lin 和 Hanson, 1994)。此外，Jaakkola、Jordan 和 Singh(1994) 在在线更新的情况下证明了趋同。所有这些结果都假设了线性无关的特征向量，这意味着 wt 的分量至少和状态一样多。对于更重要的一般 (依赖) 特征向量的收敛性首先由 Dayan(1992) 展示。Tsitsiklis 和 Van Roy(1997) 对 Dayan 的结果进行了一个重要的概括和强化。证明了本节给出的主要结果，即线性自举法渐近误差的界。

9.5 我们对线性函数逼近的可能性范围的表示是基于巴托 (1990) 的观点。

9.5.2 Konidaris, Osentoski, Thomas(2011) 在 a 中引入了傅立叶基简单的形式适用于多维连续状态空间和函数的强化学习问题，这些函数不必是周期性的。

9.5.3 粗码一词来源于 Hinton(1984)，我们的图 9.6 基于此他的一个数字。Waltz 和 Fu(1965) 在增强学习系统中提供了这种函数近似的早期例子。

238 年第 9 章: 带有近似的政策预测

9.5.4 块编码 (包括散列) 由 Albus 引入 (1971,1981)。他德- 用他的“小脑模型关节控制器”(CMAC) 来描述它，就像我们在文献中所知道的那样。“tile coding”这个术语在本书的第一版中是新的，尽管在这些术语中描述 CMAC 的想法是取自 Watkins(1989)。在许多增强学习系统中使用了 Tile coding(例如 Shewchuk 和 Dean, 1990); 林和金姆,1991; 米勒，Scalera 和 Kim, 1994;Sofge 和白色,1992;Tham,1994; 萨顿,1996; 以及其他类型的学习控制系统 (例如，Kraft 和 Campagna, 1990; 克拉夫特，米勒和迪茨，1992)。这一节主要介绍了米勒和格兰仕 (1996) 的工作。用于 tile 编码的通用软件有几种语言 (例如，请参见 http://incompleteideas.net/tiles/tiles3.html)。

9.5.5 利用径向基函数进行函数逼近得到了广泛的关注自从与 ANNs 有血缘关系，由 Broomhead 和 Lowe(1988) 著。Powell(1987) 回顾了 RBFs 的早期使用，Poggio 和 Girosi(1989, 1990) 广泛开发和应用了这种方法。

9.6 步长参数的自动调整方法包括 RMSprop (Tiele-) man and Hinton, 2012)，Adam (Kingma and Ba, 2015)，随机 meta-descent 方法，如 Delta-Bar-Delta (Jacobs, 1988)，它的增量一般化 (Sutton, 1992b, c); 和非线性推广 (Schraudolph, 1999, 2002)。明确设计用于强化学习的方法包括:AlphaBound (Dabney and Barto, 2012)、SID and NOSID (Dabney, 2014)、TIDBD (Kearney et al.，in preparation) 以及随机元下降在政策梯度学习中的应用 (Schraudolph, Yu, and Aberdeen, 2006)。

9.6 阈值逻辑单元作为抽象模型神经元的引入 McCulloch 和 Pitts(1943) 是 ANNs 的前身。ANNs 作为分类或回归学习方法的历史大致经历了几个阶段: 感知器 (Rosenblatt, 1962) 和 ADALINE (ADAptive LINear Element, Widrow and Hoff, 1960) 单层 ANNs 学习阶段，错误-反向传播阶段 (LeCun, 1985);Rumelhart, Hinton, and Williams, 1986) 的多层 ANNs 学习，以及目前以表现学习为主的深度学习阶段 (如 Bengio, Courville, Vincent, 2012);Good-

fellow, Bengio 和 Courville, 2016)。很多关于 ANNs 的书有 Haykin (1994)，Bishop(1995) 和 Ripley(2007)。ANNs 作为增强学习的函数近似，可以追溯到 Farley 和 Clark(1954) 的早期工作，他们使用强化类学习来修改代表策略的线性阈值函数的权值。Widrow, Gupta 和 Maitra(1973) 提出了一个类似神经元的线性阈值单元，该单元实现了一个学习过程，他们称之为“批评家学习”或“选择性自适应学习”，这是 ADALINE 算法的强化学习变体。Werbos(1987, 1994) 开发了一种预测和控制方法，该方法使用经过错误回溯训练的 ANNs 来学习策略和使用类 td 算法的值函数。Barto, Sutton, Brouwer(1981) 和 Barto 和 Sutton(1981) 扩展了

书目的和历史的言论 239 年

联想记忆网络的概念 (如 Kohonen, 1977;Anderson, Silverstein, Ritz，和 Jones, 1977) 强化学习。Barto、Anderson 和 Sutton(1982) 利用双层 ANN 学习非线性控制策略，强调了第一层学习合适的表示的作用。汉普森 (1983,1989) 是学习价值函数的早期支持者。Barto、Sutton 和 Anderson(1983) 以 ANN 学习平衡模拟杆的形式提出了一种角色-批评家算法 (见第 15.7 和 15.8 节)。Barto 和阿南丹 (1985) 引入了随机版本 Widrow et al。(1973) 的选择性引导算法称为关联 reward-penalty(AR−P) 算法。Barto(1985、1986) 和 Barto 和约旦 (1987) 描述了多层人工神经网络组成的基于“增大化现实”技术 −P 单位训练 globally-broadcast 强化信号学习分类规则, 并不是线性可分的。Barto(1985) 讨论了 ANNs 的这种方法，以及这种学习规则与当时文献中的其他规则之间的关系。(请参阅第 15.10 节，以进一步讨论这种培训多层人工神经网络的方法。)Anderson(1986 年、1987 年、1989 年) 对训练多层 ANNs 的多种方法进行了评估，结果表明，在河内站平衡和塔任务中，由错误反向传播训练的两层 ANNs 实现 actor - critics 算法优于单层 ANNs 算法。Williams(1988) 描述了将反向传播和强化学习结合起来进行 ANNs 训练的几种方法。Gullapalli(1990) 和 Williams(1992) 为具有连续而非二进制输出的神经元类单元设计了增强学习算法。Barto、Sutton 和 Watkins(1990) 认为，ANNs 可以在逼近解决顺序决策问题所需的函数方面发挥重要作用。Williams(1992) 相关的加强学习规则 (第 13.3 节) 对于训练多层 ANNs 的误差反向传播方法。泰索罗的 TD-Gammon(泰索罗 1992,1994;16.1 节) 有力地证明了 TD 的学习能力 ( ) 算法和函数逼近的多层人工神经网络在学习玩西洋双陆棋。Silver 等人的 AlphaGo、AlphaGo Zero、AlphaZero 程序 (2016,2017a, b); 第 16.6 节) 使用强化学习和深度卷积 ANNs 在围棋游戏中取得令人印象深刻的结果。schmidt huber(2015) 回顾了 ANNs 在强化学习中的应用，包括复发性 ANNs 的应用。

9.8 LSTD 是由于 Bradtke 和 Barto(见 Bradtke, 1993, 1994;Bradtke Barto, 1996;Bradtke,Ydstie Barto,1994), 并进一步由 Boyan(1999、2002),Nedić 和 Bertsekas(2003), 和 Yu(2010)。逆矩阵的增量更新至少从 1949 年开始就已经为人所知 (Sherman 和 Morrison, 1949)。Lagoudakis 和 Parr (2003; 总线 oniu、Lazaric Ghavamzadeh Munos, 印度绅士˘ka, 和德舒特,2012)。

9.9 我们对基于记忆的函数逼近的讨论很大程度上基于此 Atkeson、Moore 和 Schaal(1997) 对局部加权学习的回顾。Atkeson(1992) 讨论了局部加权回归在基于记忆的机器人学习中的应用，并提供了涵盖这一概念历史的广泛参考文献。Stanfill 和 Waltz(1986) 对人工智能中基于内存的方法的重要性进行了有影响的论证，特别是考虑到并行架构的出现，例如连接机。Baird 和 Klopf(1993) 提出了一种新的基于记忆的方法，并将其作为 Q-learning 的函数逼近方法应用于杆平衡任务。Schaal 和 Atkeson(1994) 将局部加权回归应用于一个机器人杂耍控制问题，并将其用于学习系统模型。Peng(1995) 利用杆位平衡任务，用几种最近邻的方法对价值函数、策略和环境模型进行了近似实验。Tadepalli 和 Ok(1996) 通过局部加权线性回归得到了有希望的结果，以学习一个模拟的自动导向车辆任务的值函数。Bottou 和 Vapnik(1992) 在某些模式识别任务中，与非本地算法相比，几个本地学习算法的效率令人惊讶，讨论了本地学习对泛化的影响。

Bentley(1975) 引入 k-d 树，并报告观察 O(log n) 的平均运行时间，对 n 个记录进行最近邻居搜索。Friedman, Bentley 和 Finkel(1977) 用 k-d 树阐明了最近邻居搜索算法。Omohundro(1987) 讨论了利用 k-d-tree 等分层数据结构可能获得的效率收益。Moore、Schneider 和 Deng(1997) 介绍了 k-d 树用于有效的局部加权回归。

9.10 内核回归的起源是 Aizerman 潜在函数的方法，布雷弗曼,Rozonoer(1964)。他们把这些数据比作在空间上分布的各种符号和大小的电荷。将点电荷的势与插值曲面相对应，在空间上产生的电势。在这个类比中，核函数是点电荷的势，它随着距离电荷的倒数而减小。康奈尔和乌特戈夫 (1987) 在杆平衡任务中应用了一种行为-批评方法，在这个任务中，批评家使用核回归和反距离加权来近似价值函数。早在对机器学习中的内核回归产生广泛兴趣之前，这些作者就没有使用内核这个术语，而是引用了“Shepard 方法”(Shepard 's method) (Shepard, 1968)。其他基于内核的强化学习方法包括 ormonit 和 Sen(2002)、Dietterich 和 Wang(2002)、Xu、Xie、Hu 和 Lu(2005)、Taylor 和 Parr(2009)、Barreto、Precup 和 Pineau(2011)、Bhat、Farias 和 Moallemi(2012)。

为了强调 td 方法，请参阅第 11.8 节的参考书目。

我们知道的最早的函数近似方法用于学习价值函数的例子是 Samuel 的 checker player(1959, 1967)。Samuel 遵循 Shannon(1950) 的建议，一个值函数不必精确到可以作为在游戏中选择动作的有用指南，它可以通过特征的线性组合来近似。除了线性函数近似外，Samuel 还试验了称为签名表的查找表和分层查找表 (Griffith, 1966, 1974; 页,1977; 比尔曼，费尔菲尔德和贝莱斯，1982 年)。Bellman 和 Dreyfus(1959) 几乎与 Samuel 的工作同时，用 DP 的函数逼近方法提出了这一观点。(人们很容易认为贝尔曼和塞缪尔对彼此产生了一些影响，但我们知道，在这两个人的作品中都没有提到对方。) 关于函数逼近方法和 DP 有相当广泛的文献，如使用样条和正交多项式的多网格方法和方法 (如 Bellman 和 Dreyfus, 1959;Bellman, Kalaba 和 Kotkin, 1973; 丹尼尔,1976;? 威特 1978;Reetz,1977; 施韦策 Seidmann,1985;Chow Tsitsiklis,1991; 库什纳 Dupuis,1992; 生锈,1996)。Holland(1986) 分类器系统使用选择性特征匹配技术在状态-动作对之间泛化评价信息。每个分类器都匹配一个状态子集，该状态为特性的子集指定了值，而其余的特性具有任意的值 (“通配符”)。然后，这些子集被用于常规的状态聚合方法来逼近函数。霍兰德的想法是使用一种遗传算法来进化一组分类器，这些分类器集体实现一个有用的动作-值函数。霍兰德的思想影响了作者对强化学习的早期研究，但我们关注的是函数逼近的不同方法。作为函数逼近器，分类器在许多方面都受到限制。首先，它们是状态聚合方法，在缩放和有效地表示平滑函数方面存在相应的限制。此外，分类器的匹配规则只能实现与特征轴平行的聚合边界。也许传统分类器系统最重要的局限性是，分类器是通过遗传算法 (一种进化方法) 来学习的。正如我们在第 1 章中讨论的那样，在学习更多关于如何学习的详细信息的过程中，我们可以用进化的方法来使用。这种观点引导我们采用监督学习方法来进行强化学习，特别是梯度下降法和神经网络方法。荷兰之间的这些差异的方法和我们并不令人惊讶, 因为荷兰的思想开发期间当 ann 被普遍认为是太弱计算能力是有用的, 而我们的工作是在一开始的时期, 普遍的质疑, 传统智慧。仍然有许多机会将这些不同方法的各个方面结合起来。Christensen 和 Korf(1986) 在国际象棋游戏中尝试了修正线性值函数近似系数的回归方法。Chapman 和 Kaelbling(1991) 和 Tan(1991) 采用决策树方法来学习价值函数。基于解释的学习方法也被用于学习价值函数，产生紧凑的表述 (Yee, Saxena, Utgoff, Barto, 1990;Dietterich 和 Flann,1995)。

10. 第十章在政策控制近似

在这一章里, 我们回到控制问题, 现在行为价值的参数近似函数 $\hat{q}$(年代,w) 问 $*$(,), 其中 w Rd 是一个有限维度权重向量。我们继续限制对政策上的案例的关注，把政策上的方法留给了第 *11* 章。本章主要介绍半梯度 *Sarsa* 算法，半梯度 *TD(0)* 在动作值和策略控制上的自然扩展。在情景性的情况下，扩展是简单的，但是在连续的情况下，我们必须后退几步，重新检查我们如何使用折现来定义最优策略。令人惊讶的是，一旦我们有了真正的函数逼近，我们就不得不放弃折现，转而使用新的“平均奖励”来制定控制问题，新的“微分”值函数。首先从情景性的例子开始，我们将上一章提出的函数逼近思想从状态值扩展到行为值。然后我们扩展他们控制政策谷歌价格指数的一般模式, 使用 ε-*greedy* 行动选择。我们在山车问题上给出了 n 阶线性萨尔萨的结果。然后我们转向继续的情况，重复这些想法的发展为平均回报情况下的差异值。

10.1 章节 Semi-gradient 控制

将第 9 章的半梯度预测方法扩展到作用值是很简单的。在这种情况下它是近似行为价值函数, 问^ q , 表示为参数化函数形式与权向量 w。而在我们考虑随机培训形式的例子圣?→Ut, 现在我们考虑的例子形成圣,?→Ut。更新目标 Ut 可以是任何近似 q (圣,), 包括常见的备份等完整的蒙特卡罗返回值 (Gt) 或任何 n-step 撒尔沙返回 (7.4)。动作值的通用梯度下降更新。

预测是

wt + 1 . = wt +

Ut− 问^(圣,wt)

wt 问^(St)。(10.1)

例如，一步 Sarsa 方法的更新是

wt + 1 . = wt +

Rt + 1 + q̂(圣 + 1 + 1,wt)− 问^(圣, 在 wt)

wt 问^(St)。(10.2)

我们称这种方法为间歇性半梯度一步 Sarsa。对于常数策略，该方法收敛的方式与 TD(0) 相同，具有相同的误差范围 (9.14)。为了形成控制方法，我们需要将这种行为价值预测方法与政策改进和行动选择技术相结合。适用于连续操作或大型离散集合的操作的合适技术，是目前尚没有明确解决方案的研究课题。另一方面，如果操作集是离散的，而且不太大，那么我们可以使用前面几章中已经开发的技术。为每个可能的行动, 在圣当前状态, 我们可以计算 q̂(圣,wt), 然后找到贪婪的行动 $*$t = argmaxa 问^(圣,wt)。政策改进然后做 (在本章在政策的情况下治疗) 通过改变估计政策的软近似贪婪 ε-greedy 政策等政策。操作是根据相同的策略选择的。在方框中给出了完整算法的伪代码。

例如，在图 10.1 左上方的图表所示，山地车任务考虑的任务是在陡峭的山路上驾驶一辆动力不足的汽车。问题在于，重力比汽车的引擎更强，即使在全速行驶时，汽车也无法加速爬坡。唯一的解决办法是先离开目标，沿着左边的相反方向往上走。然后，通过完全油门汽车

图 10.1: 山车任务 (上左面板) 和 cost-to-go 函数 (−maxa 问ˆ(年代,w)) 学会在一次运行。

它可以积累足够的惯性使它沿着陡峭的斜坡向上移动，即使整个过程都在减速。这是一个连续控制任务的简单例子，在某种意义上，事情必须变得更糟 (离目标更远)，才能变得更好。许多控制方法在处理此类任务时都有很大的困难，除非有人类设计者的明确帮助。这个问题的奖励是 −1 在所有时间步骤直到车移动过去它的目标位置在山顶, 结束这一事件。有三种可能的行为: 加足马力向前 (+ 1), 加足马力反向 (−1), 和零油门 (0)。汽车根据一个简化的物理移动。它的位置、xt 和速度,ẋt, 更新

xt + 1 。= 绑定

xt + ẋt + 1

ẋt + 1 。= 绑定

ẋt + 0.001−0.0025 cos(3 xt), 在绑定操作执行 −1.2 xt + 1 0.5−0.07 ẋt + 1 0.07。此外, 当 xt + 1 到达左绑定,ẋt + 1 是重置为零。当它到达正确的界限时，目标就实现了，这一情节就结束了。每一集开始从一个随机位置 xt (−0.6−0.4) 和零速度。为了将两个连续状态变量转换为二进制特征，我们使用了如图 9.9 所示的网格倾斜。我们使用了 8 个倾斜，每一个平铺都覆盖了每个维度上 1/8 的有界距离，第 9.5.4.1 节描述的不对称偏移量，然后将由 tile coding 生成的特征向量 x(s, a) 与参数向量线性组合，近似出动作-值函数:

问ˆ(s w)。= w ?x(,)= d ?i = 1 wi • xi(,), (10.3)

对于每一对状态 s 和动作 a。图 10.1 显示了在学习用这种形式的函数逼近来解决此任务时通常发生的情况。显示的是在一次运行中学习到的值函数 (成本-go 函数) 的负值。最初的行动值都为零, 这是乐观的 (所有真值是-在这个任务中), 导致广泛的探索发生即使勘探参数, 0。这可以在图的中上面板中看到，标记为“第 428 步”。在这个时候，甚至连一集都没有完成，但是汽车在山谷里来回摆动，沿着州空间的圆形轨迹运动。所有访问过的州都比未探索过的州的价值要低，因为实际的回报比 (不切实际的) 预期要差。这不断地将代理从它所在的任何地方赶走，以探索新的状态，直到找到一个解决方案。图 10.2 显示了在这个问题上的半梯度 Sarsa 的几个学习曲线，有不同的步长。

200 年 400 山车每集的步数平均超过 100 分

图 10.2: 半梯度 Sarsa 算法的山地车学习曲线函数逼近和 -greedy 行动选择。

特别地，我们使用了在 http://incompleteideas.net/tiles/上提供的 tile 编码软件。tiles3。使用 iht= iht(4096) 和 tiles(iht,8，[8*x/(0.5+1.2)，8*xdot/(0.07+0.07))，A) 获取状态 (x, xdot) 和动作 A 的特征向量中的索引。2 这些数据实际上是“semi-gradient 撒尔沙 ( )”算法, 我们不会满足, 直到第 12 章，但是半梯度萨尔萨的表现是相似的。

10.2 Semi-gradient n-step 撒尔沙

在半梯度 Sarsa 更新方程 (10.1) 中，我们可以使用 n 步返回作为更新目标，得到情景半梯度 Sarsa 的 n 步版本。n 阶返回立即从它的表格形式 (7.4) 推广到函数逼近形式: Gt:t + n。= Rt + 1 + Rt + 2 + • • • + n−1 Rt + n + nq̂(圣 + n + n,wt + n−1),t + n < t,Gt(10.4):t + n。= Gt 如果 t + n t, 像往常一样。n 阶更新方程是

wt + n 。= t + n−1 + (Gt:t + n− 问ˆ(圣, 在 wt + n−1)] 问ˆ(圣, 在 wt + n−1),0 t < t。(10.5)

完整的伪代码在下面的框中给出。

200 年 400 山车每集的步数平均超过 100 分

图 10.3: 单步与 8 步半梯度萨尔萨在山地车任务中的性能。好大小使用步: = 0.5/8 n = 1 和 = 0.3/8 n = 8。

280 年
山的车
240 年每集 260 步平均值前 50 集和 100 集 220 年
0 0.5 1 1.5
× 瓷砖的数量 (8)
图 10.4: 和 n 对早期性能的影响 n-step semi-gradient 撒尔沙在山地车任务中，代码函数近似。和往常一样，中级水平引导 (n = 4) 表现最好。这些结果对于选择 值, 在对数尺度, 然后用直线连接。n = 1 的标准误差范围为 0.5(小于线宽)，n = 16 的标准误差范围为 4，所以主要影响都具有统计学意义。

练习 10.1 我们还没有明确地考虑或给出任何蒙特卡罗方法的伪代码或在本章中。他们会是什么样子? 为什么不为它们提供伪代码呢? 他们将如何完成山地车任务??

练习 10.2 给出半梯度单步预期 Sarsa 的伪代码进行控制。吗? 练习 10.3 为什么图 10.4 所示的结果具有更高的标准错误大 n 比小 n 大? ?

10.3 平均奖励: 持续任务的新问题设置

我们现在介绍了第三种经典的设置——与情景性和折扣的设置——用于在马尔可夫决策问题 (MDPs) 中制定目标。和折扣设置一样，平均奖励设置也适用于持续的问题，即代理和环境之间的交互一直持续下去，没有终止或启动状态。然而，与这种设定不同的是，没有折扣——代理关心的是延迟奖励，而不是即时奖励。平均奖励设置是动态规划经典理论中普遍考虑的主要设置之一，在强化学习中不太常见。正如我们在下一节中所讨论的，贴现设置在函数逼近中存在问题，因此需要使用平均回报设置来替换它。平均报酬的设置, 政策 的质量被定义为奖励的平均利率, 或简单的平均回报, 而这一政策后, 我们表示 r( ):

= lim t→∞ E(t Rt | S0 A0:−1), (10.7)
=
s (s)
一个 (|)? r s ?, p(s ?)r,r | s,

期望以初始状态 S0 和后续动作为条件，A0 A1···据 −1, 被。 稳态分布, (年代)。圣 = = limt→∞ 公关 | A0:t−1 , 这是假定存在任何 和 S0 的独立。这种关于 MDP 的假设被称为遍历性。这意味着，在 MDP 开始的地方或由代理做出的任何早期决定都只能产生暂时的影响; 从长远来看，对处于一个国家的期望只取决于政策和 MDP 的转移概率。遍历性足以保证上述方程极限的存在。在未贴现连续情况下，不同的最优性之间可以有微妙的区别。然而, 对于大多数实用目的可能是足够简单的秩序政策根据每时间步的平均回报, 换句话说, 根据他们的 r( )。这个量是平均奖励在 , 所显示 (10.7)。特别是, 我们考虑所有的政策获得的最大价值 r( ) 是最优的。注意, 稳态分布的特殊分布, 如果您选择的行为根据 , 你留在相同的分布。这是,

s (s)
一个 (|)p(s ?| 年代,)= (s ?)。(10.8)

在平均奖励设置中，返回的定义是在不同之间。
250 年第 10 章: 带有近似的政策控制
奖励和平均奖励:

Gt。= Rt + 1 Rt + 2−−r( )+( )+ Rt + 3−r( )+ •••。(10.9) 这称为微分返回，相应的值函数称为微分值函数。他们以同样的方式定义, 我们将使用相同的符号一直:v (年代)。= E [Gt | 圣 = s] 和 q (年代)。圣 = = E [Gt | 年代, 在 =](类似 v∗ 和 q∗)。微分值函数也有 Bellman 方程，和我们之前看到的稍有不同。我们只是删除所有 s 替换所有奖励, 奖励和真正的平均回报的区别:

v =

一个 (|)? (r,s ? p(s ?r | s,)
−r( )+ v (?)
q (,)=
(r,s ? p(s ?r | s,)
−r( )+ ? 一个? (| 年代?)q (s ?, 一个?)
,
v∗(s)= max 一个
(r,s ? p(s ?r | s,)
r− 马克斯 r( )+ v∗(?)
,
问 ∗(,)=
(r,s ? p(s ?r | s,)
r− 马克斯 r( )+ 马克斯一个? 问 ∗(s ?, 一个?)
(cf.(3.14)，练习 3.17(3.19) 和 (3.20))。还有两个 TD 错误的差异形式:
t。= Rt + 1−R t + 1 + v̂(wt) 圣 + 1−v̂(St,wt), (10.10) 和 t。= Rt + 1−Rt + 1 + q̂(圣 + 1 + 1,wt)− 问ˆ(圣, 在 wt), (10.11)

在 Rt 估计在时间 t 的平均回报 R( )。有了这些替代的定义，我们的大多数算法和许多理论结果都可以在不改变的情况下进行平均回报设置。例如，半梯度 Sarsa 的平均奖励版本被定义为 (10.2)，除了 TD 错误的差分版本。也就是说, 通过

wt + 1 . q = wt + tˆ(圣, 在 wt), (10.12)

与 t 由 (10.11) 给出。完整算法的伪代码在下一页的框中给出。练习 10.4 给出半梯度 Q-learning 微分版本的伪代码。吗? 练习 10.5 需要什么方程 (超过 10.10) 来指定微分版本的 TD(0)? ?

练习 10.6 考虑一个由三个状态组成的环 a、B 和 C 组成的马尔可夫奖励过程，状态转换在环周围以确定的方式进行。到达 A 时，奖励为 +1，否则奖励为 0。微分的值是多少三个州? ?

示例 10.2: 访问控制队列任务这是一个决策任务，涉及到一组 10 台服务器的访问控制。四个不同优先级的客户到达一个队列。如果给客户访问服务器，客户将根据自己的优先级向服务器支付 1、2、4 或 8 的奖励，优先级更高的客户将支付更高的费用。在每个时间步骤中，位于队列头部的客户要么被接受 (分配给一个服务器)，要么被拒绝 (从队列中删除，奖励为零)。在这两种情况下，下一个时间步骤将考虑队列中的下一个客户。队列从不清空，队列中客户的优先级也是随机分布的。如果没有免费的服务器，当然不能为客户提供服务; 在这种情况下，客户总是被拒绝。每个繁忙的服务器在每个时间步上都有可能是 p = 0.06。虽然我们刚才已经明确地描述了它们，但是让我们假设到达和离开的统计数字是未知的。任务是根据他的优先级和免费服务器的数量，决定是否接受或拒绝下一个客户，以便在不打折扣的情况下最大化长期回报。在这个例子中，我们考虑这个问题的表格式解决方案。虽然状态之间没有泛化，但是我们仍然可以在一般的函数逼近设置中考虑它，因为这个设置泛化了表格设置。因此，我们对每一对状态 (空闲服务器数量和位于队列顶端的客户的优先级) 和操作 (接受或拒绝) 都有一个不同的动作值估计。图 10.5 显示了解决发现的微分 semi-gradient 撒尔沙与参数 = 0.01, = 0.01, = 0.1。最初的行动值和 R 是零。5

-5
-10 年
0 1 2 3 4 5 6 7 8 9 10。
免费的服务器数量

图 10.5:2 百万步后访问控制队列任务上的微分半梯度单步 Sarsa 的策略和值函数。图表右边的下降可能是由于数据不足; 许多这样的国家从未经历过。学会了 R 值¯ 约为 2.31。

假设有一个 MDP，在任何策略下产生奖励 +1、0、+1、0、+1、+1、0 的确定性序列⋯永远。严格地说，这是不允许的，因为它违反了遍历性; 没有固定的极限分布 和限制 (10.7)

并不存在。然而，平均奖励 (10.6) 是很明确的; 它是什么? 现在考虑 MDP 中的两个状态。从 A 开始，奖励序列与上面描述的完全一样，从 A +1 开始，而从 B 开始，奖励序列以 0 开始，然后以 +1、0、+1、0 继续……微分回报 (10.9) 在这种情况下没有很好的定义，因为极限不存在。要修复这个问题，可以交替地定义状态的值为

v (年代)。= lim →1 lim h→∞ h ?t = 0 t

E Rt + 1 | S0 =[s]−r( )

. (10.13)

在这个定义下，状态 A 和状态 B 的值是多少? ? 251 页上的练习 10.8 盒子里的伪代码更新 Rt + 1 使用 t 作为一个错误而不是简单的 Rt + 1−Rt + 1。这两个错误的工作, 但是使用 t 更好。要了解原因，请考虑练习 10.6 中三个状态的环 MRP。平均报酬的估计值应趋向于其真实值 1 3。假设它已经在那里并被扣留

卡在那里。什么 Rt−Rt 错误的顺序是什么? t 错误的顺序是 (使用 (10.10))? 如果允许估计随着误差的变化而变化，那么哪个误差序列会对平均奖励产生更稳定的估计? 为什么? ?

10.4 不赞成折现设置

连续的、折现的问题公式在表列情况下非常有用，在这种情况下，每个状态的回报可以分别确定和平均。但在近似情况下，人们是否应该使用这个问题公式是值得怀疑的。要知道为什么，要考虑一个没有开始或结束的无限序列，没有明确的确定状态。这些状态可能只由特征向量表示，而特征向量对于区分状态和其他状态几乎没有作用。作为特殊情况，所有的特征向量可能是相同的。因此，一个人实际上只有奖励序列 (和行为)，而绩效只能从这些行为中进行评估。怎么做呢? 一种方法是在长时间间隔内对奖励进行平均——这是平均奖励设置的思想。如何使用贴现? 每个时间步都可以衡量折现收益。有些回报是小的，有些是大的，所以我们需要在足够大的时间间隔内对它们进行平均。在连续设置中，没有开始和结束，也没有特殊的时间步骤，所以没有其他事情可以做。然而，如果你这样做，结果是折现回报的平均值与平均回报成正比。事实上, 对于政策 的平均折扣回报总是 r( )/(1−), 也就是说, 它本质上是平均回报,r( )。特别地，在平均折扣返回设置中所有策略的顺序与在平均奖励设置中是完全相同的。贴现率 因此没有影响问题公式化。它实际上可能是零，排名也不会改变。这个令人惊讶的事实在下一页的盒子里被证明了，但是基本的观点可以通过一个对称的论证来发现。每个时间步都完全相同。有了折现，每个奖励都会在每个位置出现一次。t 奖励将尚未完全出现在 t−1 日返回, 一旦在 t−2 日返回, 折扣和折扣 999 次 t−1000 返回。重量在 t 因此奖励 1 + + 2 + 3 + ••• = 1 /(1−)。因为所有的国家都是一样的, 他们都是加权, 因此回报率的平均值将它乘以平均奖励, 或者 r( )/(1−)。这个例子和盒子里更一般的论点表明，如果我们优化折现值，而不是政策上的分配，那么效果将等同于优化未折现平均回报; 的实际价值没有影响。这强烈地表明，在函数逼近控制问题的定义中，贴现没有作用。尽管如此，我们仍然可以在解决方法中使用贴现。折扣参数 参数变化从一个问题解决方法参数! 然而，不幸的是，在这种情况下，我们不能保证优化平均回报 (或在政策上分配的等效折现值)。

折现控制设置困难的根本原因是由于函数逼近，我们失去了策略改进定理 (第 4.2 节)。如果我们改变政策以提高一个国家的贴现价值，那么我们就肯定能在任何有用的意义上改进总体政策。这种保证是我们强化学习控制理论的关键。通过函数逼近，我们失去了它! 事实上，缺乏一个政策改进定理也是一种理论缺陷，这是对总情景和平均报酬设置的一种解释。一旦引入函数逼近，我们就不能保证对任何设置的改进。在第 13 章中，我们介绍了一种基于参数化策略的可选增强学习算法，在这里我们有一个理论保证，叫做“政策梯度定理”，它与政策改进定理起着类似的作用。但是对于那些学习行动价值的方法，我们目前似乎没有一个当地的改进保证 (可能是 Perkins 和 Precup(2003) 采取的方法提供了部分答案)。我们知道 -greedification 有时可能会导致劣质政策, 政策可能悄悄议论好的政策而不是收敛 (戈登,1996)。这是一个具有多个开放理论问题的领域。

10.5 差分半梯度 n 步萨尔萨

为了推广到 n 步引导，我们需要一个 n 步版本的 TD 错误。我们首先将 n 阶返回 (7.4) 推广到它的微分形式，并使用函数近似:

Gt:t + n。= Rt + 1−Rt Rt + 2−+ 1 + Rt Rt + n + 2 + •••+−Rt + n + q̂(圣 + n + n,wt + n−1), (10.14)¯ 是一个估计的 R( ),n 1, 和 t + n < t。如果 t + n t, 那么我们定义 Gt:t + n。像往常一样 = Gt。然后是 n 阶 TD 错误 t。q = Gt:t + n−^(圣, 在 w), (10.15) 之后我们可以应用我们通常的半梯度 Sarsa 更新 (10.12)。在方框中给出了完整算法的伪代码。

10.9 运动微分 semi-gradient n-step 撒尔沙算法步长参数的平均回报, , 需要非常小所以 R 成为一个良好的长期的平均估计奖励。不幸的是,R 将偏见的初始值为许多步骤, 这可能会使学习效率低下。或者, 可以使用一个样本的平均 R, 观察到的报酬。这将在最初迅速适应，但从长远来看也将缓慢适应。随着政策慢慢改变,R 也会变化; 这种长期的非平稳性的可能性使得样本平均方法不适合。实际上，平均奖励上的步长参数是使用练习 2.7 中无偏恒步长技巧的完美地方。描述用于差分半梯度 n 阶 Sarsa 的装箱算法所需的具体变化技巧。?

10.6 总结

在本章中，我们将前一章介绍的参数化函数逼近和半梯度下降的思想扩展到控制。对情节的扩展是直接的，但是对于连续的情况我们必须引入一个全新的问题公式基于最大化每个时间步的平均奖励设置。令人惊讶的是，在近似的情况下，折现公式不能用于控制。在近似情况下，大多数策略不能用一个值函数表示。任意的政策仍然需要排名, 和标量平均奖励 r( ) 提供了一种有效的方法。平均奖励公式包括新的微分版本的值函数，Bellman 方程和 TD 错误，但所有这些都与旧的相似，概念上的变化很小。对于平均奖励情况，还有一组新的并行微分算法。

10.7 书目的和历史的言论

10.1 函数逼近半梯度 Sarsa 是 Rummery 首先探索的和 Niranjan(1994)。线性 semi-gradient 撒尔沙 -greedy 行动选择没有通常意义上的收敛, 但进入一个有界区域附近的最佳解决方案 (戈登,1996 a,1996)。Precup 和 Perkins(2003) 在可微行为选择的背景下表现出收敛性。另见珀金斯和潘德里斯 (2002)、梅洛、米恩和里贝罗 (2008)。山地车的例子是基于 Moore(1990) 研究的一个类似的任务，但是这里使用的确切形式来自 Sutton(1996)。10.2 章节 n-step semi-gradient 撒尔沙是基于向前撒尔沙 ( ) 算法的范 Seijen(2016)。本文第二版的实证结果是新的。

10.3 描述了动态规划的平均报酬公式 (例如，Puterman, 1994) 和从强化学习的观点来看 (Ma-hadevan, 1996;Tadepalli 和好的,1994;Bertsekas Tsitiklis,1996;Tsitsiklis 和 Van Roy, 1999)。这里描述的算法是 Schwartz(1993) 引入的“R-learning”算法的策略模拟。R-learning 这个名字可能是 Q-learning 的字母继承，但我们更愿意把它当作学习微分或相对值的参考。的访问控制队列的例子是由工作 Carlström 和 Nordström(1997)。

10.4 认识到贴现作为约束的一种形式的局限性在本文第一版出版后不久，函数逼近的学习问题就显现出来了。Singh、Jaakkola 和 Jordan(1994) 可能是第一个用印刷体来观察它的人。

11. 第十一章 Off-policy 方法近似

本书从第 5 章开始，对政策和非政策的学习方法进行了处理，主要是两种不同的方法来处理在学习形式的广义政策迭代中所固有的开发与探索之间的冲突。前两章用函数逼近的方法来处理保单案例，本章用函数逼近的方法来处理保单案例。对函数逼近的扩展结果表明，与对政策学习的扩展相比，偏离政策学习的扩展结果是显著不同的，也更加困难。在第 6 章和第 7 章中开发的表外策略方法很容易扩展到半梯度算法，但是这些算法不像在策略训练中那样强收敛。在本章中，我们探讨了收敛问题，深入研究了线性函数逼近理论，引入了可学习性的概念，然后讨论了具有更强收敛保证的新算法。最后，我们将改进方法，但理论结果不会像政策学习那样强，实证结果也不会像政策学习那样令人满意。在此过程中，我们将进一步加深对策略学习和非政策学习的强化学习的理解。回想一下，在 *off-policy* 学习我们寻求学习目标政策 值函数，给定数据由于不同行为策略 b。在预测的情况下，政策都是静态的，因为，我们寻求学习状态值 $\hat{v} \approx v$ 或行动值 $\hat{q} \approx q$。在控制的情况下，动作值，和这两项政策通常改变 *learning-* 期间被贪婪的政策对 $\hat{q}$，和 b 被更探索性如 ε*-greedy* 政策对 $\hat{q}$。非政策学习的挑战可以分为两部分，一个是在表格案例中出现的，另一个是在函数逼近的情况下产生的。挑战的第一部分涉及更新的目标 *(*不要与目标策略混淆*)*，第二部分涉及更新的分发。第 5 章和第 7 章中关于重要性抽样的技术涉及第一部分*;* 这些可能会增加方差，但在所有成功的算法中都需要，表格和近似。将这些技术扩展到函数逼近，将在本章的第一节中快速讨论。由于在非政策情况下的更新的分布不符合政策的分配，所以在功能近似下的非政策学习的第二个部分需要更多的东西。政策上的分布对半梯度法的稳定性有着重要的影响。已经探讨了两种一般方法来处理这一问题。一是再次使用重要的抽样方法，这一次将更新分布重新分配到 *on-policy* 分布，从而保证半梯度方法收敛 *(*在线性情况下*)*。另一种方法是开发真正的梯度方法，不依赖任何特殊的分布来获得稳定性。我们提出了基于这两种方法的方法。这是一个前沿的研究领域，目前还不清楚哪种方法在实践中最有效。

11.1 Semi-gradient 方法

我们首先描述在较早的章节中为非策略案例所开发的方法如何容易地扩展为半梯度方法。这些方法解决了脱机策略学习的第一部分 (更改更新目标)，而不是第二部分 (更改更新分布)。因此，这些方法在某些情况下可能会发散，在这种意义上不是声音，但它们仍然经常被成功地使用。记住，这些方法对于表格情形是稳定且渐进无偏的，它对应于函数逼近的一个特殊情形。因此，仍有可能将它们与特征选择方法结合在一起，从而确保组合系统的稳定性。无论如何，这些方法都很简单，因此是一个很好的起点。在第 7 章中，我们描述了各种表外策略算法。将其转换成 semi-gradient 形式，我们只是替换更新数组 (V 或 Q) 更新权重向量 (w)，使用近似值函数 (V 或 Q) 及其梯度。这些算法中有许多采用了每步重要性抽样比率:

例如，一步，州值算法 semi-gradient off-policy TD(0)，就像相应的政策算法 (第 203 页) 除了 t 的添加:

wt + 1 . = wt + t t v̂(St,wt), (11.2) t 在哪里定义适当取决于问题是情景和折扣, 或者继续和尚未完全使用平均奖励:

t。= Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt), 或 (11.3)

t。= Rt + 1−Rt + v̂(wt) 圣 + 1−v̂(St,wt)。(11.4)

11.1。Semi-gradient 方法 259 年

对于动作值，一步算法为半梯度期望 Sarsa:

wt + 1 . q = wt + t(圣, 在 wt) (11.5)

t。= Rt + 1 +

一个 (| 圣 + 1) 问(圣 + 1,wt)− 问(圣, 在 wt) 或 (情景)

t。= Rt + 1−Rt +

一个 (| 圣 + 1) 问(圣 + 1,wt)− 问(圣, 在 wt)。(继续)

注意，该算法不使用重要抽样。在列表的情况下，很明显，这是适当的，因为唯一的示例操作是 At，并且在学习它的值时，我们不需要考虑任何其他操作。对于函数逼近，情况就不那么清楚了，因为我们可能想要对不同的状态-动作对施加不同的权重，一旦它们都对相同的整体逼近做出贡献。这个问题的正确解决需要在强化学习中更深入地理解函数逼近理论。在这些算法的多步推广中，状态值算法和动作值算法都涉及到重要性抽样。例如，半梯度的 n 阶版本期望 Sarsa 是。

wt + n . = t + n−1 + t + 1 ••• t + n−1(Gt:t + n− 问(圣, 在 wt + n−1)] 问(圣, 在 wt + n−1) (11.6) 与

Gt:t + n。= Rt + 1 + ••• + n−1 Rt + n + nq̂(圣 + n + n,wt + n−1), 或 (情景)Gt:t + n。= Rt + 1−Rt Rt + n + ••• +−Rt + n−1 + q̂(圣 + n + n,wt + n−1),(继续) 这里我们对情节结尾的处理有点不正式。在第一个方程中, ks k T(T) 是最后一集的时间步) 应采取是 1,n 和 Gt: 应采取 Gt 如果 T + n T。回想一下，我们也在第七章中提出了一种不涉及重要抽样的非策略算法:n 步树备份算法。这里是它的半梯度版本:

wt + n 。= t + n−1 + (Gt:t + n− 问(圣, 在 wt + n−1)] 问(圣, 在 wt + n−1),(11.7)

Gt:t + n。= 问(圣, 在 wt−1)+ t + n−1 ?k = t k k ? 我 = t + 1 (Ai | Si), (11.8)

与 t 定义为预期的撒尔沙这个页面的顶部。我们也在第七章中定义一个算法, 结合所有行为价值算法:n-step Q( )。我们将该算法的半梯度形式，以及 n 步状态值算法，作为读者的练习。练习 11.1 将 n 步偏离策略 TD(7.9) 的方程转化为半梯度形式。给出情景和持续案例的回报的相关定义。吗?∗11.2 运动的方程转换 n-step Q( )(7.11 和 7.17)semi-gradient 形式。给出涵盖情景和连续情况的定义。?

11.2 非政策分歧的例子。

在这一节中，我们将从函数逼近的角度来讨论离线学习的挑战的第二部分——更新的分布与策略上的分布不匹配。我们描述了一些针对非策略学习的有指导意义的反例——半梯度和其他简单算法是不稳定和发散的。要建立直觉，最好先考虑一个非常简单的例子。想, 也许作为一个更大的 MDP 的一部分, 有两种状态的估算值的函数形式 w 和 2 w, 那里只有一个组件的参数向量 w 由 w。这发生在线性函数近似如果两种状态的特征向量都是简单的数字 (单组分向量), 在这种情况下 1 和 2。在第一个状态中，只有一个动作可用，并且它在向第二个状态的转换中确定的结果为 0: 2 w 0 2 w 两个圆圈内的表达式表示两个状态的值。假设初始 w = 10。然后，这个过渡将从估计价值 10 的状态变为估计价值 20 的状态。它看起来是一个很好的过渡，w 将被增加以提高第一个状态的估计值。如果 几乎是 1, 那么 TD 错误将是近 10 w, 如果 = 0.1, 那么将增加到近 11 试图减少 TD 的错误。然而，第二个州的估计价值也将增加到将近 22 个。如果再次发生转变, 那么它将从一个状态估计价值 22 日 11 状态的估计价值的 TD 错误 11-larger, 不是小, 比以前。看起来更像第一个状态被低估, 其价值将再次增加,

这时间 12.1。这看起来很糟糕，实际上随着进一步的更新，w 将会发散到无穷大。要明确地看到这一点，我们必须更仔细地观察更新的顺序。在两个状态之间的转换上的 TD 错误是

t = Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt)= 0 + 2wt−wt =(2 −1)wt, 和 off-policy semi-gradient TD(0) 更新 (从 (11.2)) wt + 1 = wt + t t v̂(St,wt)= wt + • 1 • (2 −1)wt • 1 = ? 1 + (2 −1)wt。注意, 重要性抽样比率, t, 这种转变是 1, 因为只有一个动作可以从第一个状态, 所以它的概率下的被目标和行为的政策必须是 1。在最后的更新, 新的参数是旧的参数乘以一个标量常数,1 + (2 −1)。如果这个常数大于 1, 则系统不稳定和 w 将积极或消极的无穷根据其初始值。这个常数大于 1 时 > 0.5。注意, 稳定性不依赖于具体的步长, 只要 > 0。更小或更大的步长会影响 w 趋于无限大的速率，但不会影响它是否达到无限大的速率。这个示例的关键是，一个转换重复地发生，而没有在其他转换上更新 w。这在政策外的培训中是可能的，因为

行为策略可以选择目标策略永远不会选择的其他转换上的操作。对于这些转换, t 将是零, 没有更新。在政策培训, 然而, t 总是一个。每次有一个从 w 状态到 2w 状态的转变，增加 w，也会有一个从 2w 状态的转变。过渡会降低 w, 除非它是一个国家, 其价值高 (因为 < 1) 比 2 w, 然后国家必须紧随其后的更高的价值, 否则再次 w 将会降低。每个状态只能通过创建更高的期望来支持一个状态。终有一天，风笛手必须得到报酬。在按政策执行的情况下，必须遵守未来奖励的承诺，并控制系统。但在非政策的情况下，可以做出承诺，然后在采取目标政策永远不会采取的行动之后，就会忘记和原谅。这个简单的例子说明了为什么非政策培训会导致分歧，但它并不能完全令人信服，因为它还没有完成——它只是完整的 MDP 的一部分。真的有一个不稳定的完整系统吗? 一个简单而完整的散度例子是 Baird 的反例。考虑图 11.1 所示的情景七态双作用 MDP。虚线运动使系统有相等的概率到达六种上状态之一，而实线运动使系统达到七种状态。行为策略 b 选择概率为 6 7 和的虚线和实线动作 1 因此它下的下一个状态分布是均匀的对于所有非终端状态也是一样，这也是每一集的开始分布。目标政策总是需要坚实的行动, 所以在政策分布 ( ) 集中在第七状态。所有转变的回报都是零。贴现率是 = 0.99。考虑在每个状态圆中表示的线性参数化下估计状态值。例如，最左侧状态的估计值是 2w1+w8，其中下标对应于

图 11.1:Baird 的反例。这个马尔可夫的近似状态值函数过程是每个状态中的线性表达式所显示的形式。坚实的行动通常结果是第七种状态，而虚线运动通常会导致另外六种状态中的一种，每种状态的概率都是相等的。报酬总是零。

整体权向量 w R8; 这对应于第一个状态为 x(1) =(2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,1) 的特征向量。奖励在所有转换为零, 所以真正价值函数是 v (s)= 0, 所有的年代, 可以完全近似如果 w = 0。事实上, 有许多解决方案, 作为权向量有多个组件 (8) 比非终结符 (7)。此外, 特征向量的集合, x(s) 的模式:s 年代, 在所有这些方面都是一个线性无关组。这个任务似乎是一个良好的线性函数近似。如果我们对这个问题 (11.2) 应用半梯度 TD(0)，那么权值会发散到无穷大，如图 11.2(左) 所示。不稳定性发生在任何正的步长，无论多小。事实上，如果像 dynamic programming (DP) 那样进行预期的更新，甚至会发生这种情况，如图 11.2 所示 (右)。也就是说，如果权重向量 wk 同时以半梯度的方式对所有状态进行更新，使用 DP(基于期望) 目标:

周 + 1 . = 周 + | 的 |

s

E (Rt + 1 + v̂(圣 + 1 周)| 圣 = s]−v̂(年代, 周)

v̂(年代, 周)。(11.9)

在这种情况下，不存在随机性和异步性，就像经典的 DP 更新一样。除了使用半梯度函数近似外，该方法是常规的。然而，这个体系仍然不稳定。如果我们在 Baird 的反例中仅仅改变 DP 更新的分布，从均匀分布到策略内分布 (通常需要异步更新)，那么可以保证收敛到一个误差为 (9.14) 的解决方案。这个例子引人注目，因为所使用的 TD 和 DP 方法可以说是最简单的将 One hundred.

10 1

1000 0 的步骤
将
1000 0 扫

图 11.2:Baird 反例的不稳定性证明。展示了两种半梯度算法参数向量 w 的分量的演化。步长 = 0.01, 和初始重量 w =(1,1,1,1,1,1,1) 吗?。

而最容易理解的自举法，以及所使用的线性、半下降法，可以说是最简单、最容易理解的一类函数逼近。这个例子表明，即使是自举和函数逼近的最简单的组合也可能不稳定，如果更新不是按照 on-policy 的分布进行的。也有类似 Baird 的反例显示出 Q-learning 的发散。这是令人担忧的原因，否则 Q-learning 具有所有控制方法的最佳收敛性保证。为了找到解决这一问题的办法或得到一些较弱但仍然可行的保证，已经作出了相当大的努力。例如, 它可能会保证收敛的 q 学习只要行为政策是足够接近目标政策, 例如, 当它是 -greedy 政策。就我们所知，在这种情况下，Q-learning 从未被发现有分歧，但还没有理论分析。在本节的其余部分中，我们将介绍一些已经探讨过的其他想法。假设我们不是像 Baird 的反例那样，在每次迭代中向预期的单步返回迈出一步，而是将值函数一直更改为最佳的最小二乘逼近。这会解决不稳定问题吗? 当然如果特征向量, x(s) 的模式:s 年代, 形成一个线性无关组, 在 Baird 的反例, 因为精确的近似是可能的在每个迭代和表格 DP 的方法降低标准。但这里的重点当然是考虑当不可能得到精确解的情况。在这种情况下，即使在每次迭代中形成最佳逼近时，稳定性也得不到保证，如示例中所示。示例 11.1:tsiklis 和 Van Roy 的反例本例表明，即使最小二乘，线性函数逼近也不能用于 DP

1−
w 2 w 每一步都有解决方案。反例是通过扩展 wto -2w 示例 (来自本节前面的部分) 来形成的，该示例具有末端状态，如右边所示。如前所述，第一个状态的估计值为 w，第二个状态的估计值为 2w。所有转变的回报都是零，所以两种状态的真实值都是零，这在 w = 0 时是完全可以表示的。如果我们在每一步都设置 wk+1，以最小化估计值和期望的单步返回之间的 VE，那么我们就得到了

周 + 1 = argmin w R
年代
v̂(s,w)−E 吗?Rt + 1 + v̂(圣 + 1 周)? 圣 = 年代? 2 = argmin w R
w− 2wk 2 + 2 w ?−1−) 2wk 2
= 6−4 5 wk。(11.10) 序列周发散当 > 5 6−4 ,w0 = 0。

另一种防止不稳定性的方法是使用函数逼近的特殊方法。特别是，函数逼近方法不从观测目标外推的稳定性得到了保证。这些方法，被称为平均值，包括最近的邻居方法和局部加权回归，但不流行的方法，如 tile 编码和人工神经网络 (ANNs)。练习 11.3(编程) 应用一步半梯度 q 学习到 Baird 的 coun-。举个例子来说明它的权重是不同的。?

11.3 致命的三合会

我们到目前为止的讨论可以总结为，当我们把以下三个要素结合在一起时，就会产生不稳定和分歧的危险，这就是我们所说的致命三位一体:

函数逼近是一种强大的、可扩展的方法，可以从比内存和计算资源 (例如，线性函数逼近或 ANNs) 大得多的状态空间进行归纳。

引导更新目标，包括现有的估计 (如动态规划或 TD 方法)，而不是仅仅依赖于实际的奖励和完整的回报 (如 MC 方法)。

政策外的培训，关于转变的分配，而不是目标政策产生的。遍历状态空间并统一更新所有状态，就像在动态规划中一样，不尊重目标策略，这是场外政策培训的一个例子。

特别要注意的是，危险不是由于控制或泛化策略迭代造成的。这些情况分析起来比较复杂，但在更简单的预测情况中，只要包含致命三元组的所有三个元素，就会出现不稳定性。危险

也不是由于学习或环境的不确定性，因为它在规划方法 (例如动态规划) 中同样发生，在动态规划中，环境是完全已知的。如果死亡三位一体中的任何两个元素存在，但不是全部，那么就可以避免不稳定。因此，很自然地，通过这三个，看看是否有任何一个可以被放弃。在这三种方法中，函数逼近显然是不能放弃的。我们需要的方法可以扩展到大问题，并具有强大的表达能力。我们至少需要具有许多特征和参数的线性函数近似。状态聚合或非参数方法，其复杂性随着数据的增长而增长，这些方法要么太弱，要么太昂贵。像 LSTD 这样的最小二乘方法具有二次复杂度，因此对于大型问题来说代价太高。以计算和数据效率为代价，不进行引导是可能的。也许最重要的是计算效率的损失。蒙特卡罗 (非 bootstrapping) 方法需要内存来保存生成过程中发生的所有事情。

每一个预测和得到最终的回报，所有的计算都是在得到最终的回报之后完成的。这些计算问题的成本在冯 • 诺依曼系列计算机上并不明显，但将在专用硬件上体现出来。使用 bootstrapping 和资格跟踪 (第 12 章)，数据可以在何时、何地生成，然后再也不需要使用。通过 bootstrapping 实现的通信和内存节省是很大的。放弃自举操作在数据效率上的损失也很严重。我们在第七章 (图 7.2) 和第九章 (图 9.2) 中多次看到过这种情况，在这些章节中，在随机行走预测任务中，某种程度的引导比蒙特卡罗方法表现得更好，在第十章中，在登山车控制任务中也看到了这种情况 (图 10.4)。许多其他的问题显示了自举学习的速度快得多 (例如，参见图 12.14)。自举通常会导致更快的学习，因为它允许学习利用状态属性，即返回状态时识别状态的能力。另一方面, 引导可以削弱学习问题的状态表示很差, 导致可怜的泛化 (例如, 这似乎是在俄罗斯方块, 看到 imşek, 藻类´ 等, 和 Kothiyal,2016)。糟糕的状态表示也会导致偏见; 这就是自举法渐近逼近质量较差的原因 (公式 9.14)。总的来说，引导的能力必须被认为是极有价值的。一个有时可能会选择不使用它通过选择长 n-step 更新 (或大型引导参数, 1; 参见第 12 章)，但是引导通常会极大地提高效率。这是我们非常希望保留在我们的工具箱中的一种能力。最后，还有政策外的学习; 我们能放弃吗? 政策上的方法通常是足够的。对于无模型强化学习，可以简单地使用 Sarsa 而不是 Q-learning。策略外方法将行为从目标策略中释放出来。这可以被认为是一种吸引人的便利，但不是必要的。但是，脱机学习对于其他预期的用例是必不可少的，这些用例我们在本书中还没有提到，但是对于创建强大的智能代理的更大目标可能是重要的。在这些用例中，代理不仅学习一个值函数和一个策略，还同时学习大量的值函数和策略。有大量的心理学证据表明人类和动物学会预测许多不同的感觉事件，而不仅仅是奖励。我们会对不寻常的事件感到惊讶，并纠正我们对它们的预测，即使它们是中性价 (既不好也不坏)。这种预测可能是世界的预测模型的基础，如在规划中使用的模型。我们预测我们的眼睛运动后会看到什么，走路回家需要多长时间，篮球跳投的可能性，以及接受一个新项目的满足感。在所有这些情况下，我们想要预测的事件取决于我们的行为方式。要学习它们，就需要从经验中学习。有许多目标策略，因此一个行为策略不能等同于所有的策略。然而，并行学习在概念上是可能的，因为行为策略可能部分地与许多目标策略重叠。要充分利用这一点，需要学习政策以外的知识。

11.4 线性值函数几何

为了更好地理解非策略学习的稳定性挑战，更抽象地、独立地思考价值函数逼近，而不是学习是如何完成的，这是很有帮助的。我们可以想象所有可能的状态值的空间从各州实数 v 函数功能:S→r . 这些价值函数并不对应任何政策。对于我们的目的来说，更重要的是，大多数参数不能用函数逼近器来表示，而函数逼近器的参数要比有状态的参数少得多。给定状态空间 S = s1, s2，⋯，s| s|，任何值函数 v 都对应一个向量，该向量按照 v(s1)、v(s2)、⋯v(s | |)]?。这个值函数的向量表示具有状态的分量。在大多数情况下，我们想要使用函数逼近，这将是太多的分量来显式地表示向量。然而，这个向量的概念在概念上是有用的。在下面，我们将一个值函数和它的向量表示互换。为了发展直觉，考虑三个状态的情况，分别为:s1, s2, s3 和两个参数 w = (w1, w2)。然后我们可以把所有的值函数/向量看作三维空间中的点。这

些参数在二维子空间上提供了一个可选的坐标系。有权向量 w = (w1, w2) 吗? 是二维子空间中的一个点，因此也是一个完整的值函数 vw，它将值赋给所有三个状态。利用一般函数逼近，可表示函数的全空间与子空间之间的关系可以是复杂的，但在线性值函数逼近的情况下，子空间是一个简单的平面，如图 11.3 所示。现在考虑一个固定政策 。我们假设它真正的价值函数,v , 太复杂就像一个近似表示。因此 v 不是子空间; 在图中，它被描述为在可表示函数的平面子空间之上。如果 v 不能代表完全, 可表示的值函数是最接近吗? 这是一个有多个答案的微妙问题。首先，我们需要测量两个值函数之间的距离。给定两个 v1 和 v2 价值函数, 我们可以讨论向量之间的区别,v = v1−v2。如果 v 很小，那么这两个值函数是很接近的。但是我们如何测量这个差向量的大小呢? 传统的欧几里得准则是不合适的，因为正如第 9.2 节所讨论的，有些国家比其他国家更重要，因为它们发生得更频繁，或者因为我们对它们更感兴趣 (第 9.11 节)。在 9.2 节中, 让我们使用 分布:S→[0,1] 来指定我们关心的程度不同的州是准确值 (通常采取政策分布)。然后我们可以用范数定义值函数之间的距离

v ?2 。=

年代 v(s)2。(11.11)

注意, 已经从 9.2 节可以写简单地使用这个标准已经 (w)= ? 大众 −v 吗?2 。对于任何值函数 v，在可表示值函数的子空间中寻找最接近的值函数的操作是一个投影运算。我们定义了一个

图 11.3: 线性值函数逼近的几何形式。如图所示是三所有值函数在三种状态下的维数空间，而平面表示的是所有值函数的子空间，由参数 w = (w1, w2) 的线性函数逼近器表示。真正价值函数 v 是在更大的空间, 可以预计 (子空间, 利用投影算符 Π) 的最佳逼近值错误 (VE) 意义。在 Bellman 错误 (BE)、投射 Bellman 错误 (PBE) 和时态中最好的近似器差分误差 (TDE) 的感觉都可能是不同的，显示在右下角。(VE, BE，和 PBE 都被视为这个图中对应的向量。) 更夫操作符取平面内的值函数到平面外的值函数，然后将其投影回来。如果迭代地将 Bellman 运算符应用到空间之外 (如上面的 gray 所示)，就会达到真正的值函数，就像传统的动态编程那样。如果你保持在每个步骤中向后投影到子空间中，就像在灰色显示的较低的步骤中，然后是固定的点是向量零 PBE 的点。

投影算符 Π, 需要一个任意值函数是最规范的可表示的函数:

Πv。= vw = argmin w Rd v− 大众?2 。(11.12) 能上演的价值函数, 因此是最接近真值函数 v 投影,Πv , 显示如图 11.3 所示。这是蒙特卡罗方法渐近发现的解，虽然通常很慢。投影操作将在下一页的框中更详细地讨论。TD 方法找到不同的解决方案。理解他们的理由, 回想一下, 贝尔曼方程函数 v 价值

v =

一个 (|)? r s ?, p(s ?,r | s)[r + v (?)], 所有年代 s(11.13)

真正价值函数 v 是唯一的价值函数, 解决了 (11.13)。如果一个近似值函数代替 v 大众, 左右两侧的区别修改方程可以作为衡量如何从 v 远离大众。我们称之为 s 州的 Bellman 错误:

¯w(s)。= 吗? 一个 (|)? r s ?, p(s ?,r | s)[r + vw(s ?) − 大众 (s)(11.17)

= E

Rt + 1 + vw(圣 + 1)− 大众 (St) 吗? 圣 = s,

(11.18)

它清楚地显示了 Bellman 错误与 TD 错误 (11.3) 的关系。Bellman 错误是 TD 错误的期望。向量的更夫错误, 在所有国家,¯w R | |, 叫做传达员误差向量 (如图所示, 在图 11.3)。这个向量的总体大小，在范数中，是值函数误差的总体度量，称为均值平方 Bellman 误差:

(w)= ??¯w ?2 ?。(11.19) 一般是不可能减少是零 (此时大众 = v), 但对于线性函数近似是一个独特的有价值的 w 是最小化。这一点在子空间可表示的函数 (图 11.3) 中的标记 minBE 不同一般, 这样可以最大限度减少 Πv (如图所示)。在接下来的两部分中，我们将讨论一些方法来最小化这些问题。更夫误差矢量图 11.3 所示的结果应用传达员运营商 B :R S | |→R S | | 近似值函数。更夫操作符是

定义为

(B v)(s)。=

一个 (|)? r s ?, p(s ?,r | s)[r + v(?)], (11.20)

所有 年代和 v:s→r . v 的更夫误差向量可以写¯w = B vw− 大众。如果 Bellman 算子作用于可表示子空间中的值函数，然后，一般来说，它将产生一个新的值函数，该函数位于子空间之外，如图所示。在动态规划 (没有函数逼近) 中，这个算子被反复地应用到可表示空间之外的点上，如图 11.3 顶部的灰色箭头所示。最终这一过程收敛于真值函数 v , 唯一不动点的更夫运营商唯一值函数

v = B v , (11.21) 这是另一种写法是 的贝尔曼方程 (11.13)。然而，对于函数逼近，位于子空间之外的中间值函数是无法表示的。图 11.3 上部的灰色箭头不能跟随，因为在第一次更新 (黑线) 之后，值函数必须被投影到可表示的东西上。下一个迭代从子空间开始; 值函数再次被 Bellman 算子提取到子空间之外，然后被投影算子映射回来，如下面的灰色箭头和直线所示。跟随这些箭头的是一个具有近似的 dp 过程。在这种情况下，我们感兴趣的是 Bellman 错误向量的投影回到可表示空间。这是预计传达员误差向量 Π¯ 大众,PBE 如图 11.3 所示。这个向量的大小，在范数中，是近似值函数误差的另一个度量。对于任何近似值函数 v，我们定义均方投影贝尔曼误差，记为 PBE

PBE(w)= ?w ?Π¯ 吗?? 2 。(11.22) 对于线性函数逼近，总是存在一个具有零 PBE 的近似值函数 (在子空间内); 这是 TD 固定点，wTD，在第 9.4 节介绍。正如我们所看到的，在半梯度 TD 方法和非政策训练下，这一点并不总是稳定的。如图所示，这个值函数通常与最小化 VE 或 BE 的值函数不同。保证收敛的方法将在第 11.7 和 11.8 节中讨论。

在 Bellman 错误中有 11.5 梯度下降。

在更好地理解价值函数逼近及其各种目标的基础上，我们现在回到了离线学习中稳定性的挑战。我们想应用随机梯度下降的方法 (SGD，第 9.3 节)，其中更新了期望等于目标的负梯度

函数。这些方法在目标中总是会下降 (在期望中)，因为它具有典型的稳定性，具有很好的收敛性。在本书所研究的算法中，只有蒙特卡罗方法是真正的 SGD 方法。这些方法在政策和非政策训练以及一般的非线性 (可微) 函数逼近器下都是鲁棒收敛的，尽管它们通常比自举的半梯度方法慢，而非 SGD 方法。半梯度方法在偏离策略的训练下可能会有分歧，正如我们在本章前面看到的，在设计的非线性函数逼近的情况下也会有分歧 (tsiklis 和 Van Roy, 1997)。用真正的 SGD 方法，这种发散是不可能的。SGD 的吸引力是如此之强，以至于巨大的努力已经开始寻找一种实用的方法来利用它来加强学习。所有这些努力的出发点都是选择一个要优化的错误或目标函数。在本节和下一节中，我们将探讨基于前一节中引入的 Bellman 错误的最流行的目标函数的起源和限制。尽管这是一种流行的、有影响力的方法，但我们得出的结论是，这是一个错误的步骤，没有好的学习算法。另一方面，这种方法以一种有趣的方式失败了，它提供了对什么可能构成一个好的方法的洞察。首先，让我们不要考虑 Bellman 错误，而是考虑更直接、更天真的问题。时间差异学习是由 TD 误差驱动的。为什么不把 TD 误差的期望平方的最小化作为目标呢? 在一般的函数逼近情况下，带有折扣的单步 TD 误差为

t = Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt)。

那么，一个可能的目标函数就是所谓的平均平方 TD 误差:TDE(w) =? 年代 (s)E ? 2 t ? 圣 = s, =

年代 (s)E ? t 2 t ? 圣 = s, b = Eb

t 2 t

。(如果 b) 下遇到的分布

最后一个方程是 SGD 所需的形式; 它将目标作为可以从经验中抽取的期望 (请记住，经验是由于行为策略 b)。因此，遵循标准的 SGD 方法，可以根据该期望值的一个样本得出每步更新:

wt + 1 =− 那样 1 2 (t 2 t) = − t t t = wt + t t

v̂(圣 wt)− v̂(wt) 圣 + 1, (11.23) 除了附加的最后一项之外，你会发现它和半梯度 TD 算法 (11.2) 是一样的。这一项完成了梯度，使它成为一个真正的 SGD 算法，具有极好的收敛

保证。我们把这个算法叫做朴素算法

残差梯度算法 (Baird 之后，1995)。虽然原始的残差梯度算法是鲁棒收敛的，但它不一定收敛到理想的位置。

在 A-split 示例中使用表格表示，因此可以精确地表示真实的状态值，但是原始的 residual-gradient 算法会找到不同的值，并且这些值的 TDE 低于真实值。最小化 TDE 是幼稚的; 通过惩罚所有 TD 错误，它实现了时间平滑而不是准确的预测。更好的办法似乎是最小化 Bellman 错误。如果知道了确切的值，那么 Bellman 错误在任何地方都是零。因此，bellman -error 最小化算法对于 a -split 示例应该没有问题。我们不能期望得到一般的零 Bellman 错误，因为它涉及到找到真值函数，我们假定它在可表示值函数的空间之外。但接近这个理想是一个看似自然的目标。正如我们所看到的，Bellman 错误也与 TD 错误密切相关。一个状态的 Bellman 错误是该状态的预期 TD 错误。让我们重复前面的推导，得到期望的 TD 误差 (这里所有的期望都隐式地以 St 为条件):

wt + 1 =− 那样 1 2 (E (t)2)

= − 1 2 (Eb(t t)2) = − Eb[t t] Eb[t t]= − Eb

t(Rt + 1 + v̂(w) 圣 + 1−v̂(St,w))eb(t t] = wt +

海尔哥哥

t(Rt + 1 + v̂(w) 圣 + 1)−v̂(St,w)

v̂(圣,w)− Eb ? t v̂(圣 + 1 w)

这种更新和各种采样方法被称为残差梯度算法。如果您只是在所有期望中使用了样本值，那么上面的方程几乎完全可以简化为 (11.23)，即原始的 residual-gradient 算法。但这很天真，因为上面的方程包含了下一个状态 St+1，出现在两个期望中，然后相乘。要获得产品的无偏样本，需要两个下一个状态的独立样本，但在与外部环境的正常交互过程中，只获得一个。一个期望或另一个期望可以被抽样，但不能同时抽样。有两种方法可以使 residual-gradient 算法工作。一个是确定性环境。如果到下一个状态的转换是确定的，那么两个样本必然是相同的，并且朴素算法是有效的。另一种方法是从 St 中获得下一个状态的两个独立样本 St+1，一个是第一个期望，另一个是第二个期望。在与环境的实际交互中，这似乎是不可能的，但是当与模拟环境交互时，这是可能的。一个简单地回滚到以前的状态，然后在从下一个状态继续前进之前获得另一个状态。在这两种情况下，都保证了在通常的步长参数条件下，残差梯度算法收敛到最小值。作为一种真正的 SGD 方法，这种收敛是

对于状态值，在处理重要性抽样比时，仍然存在很小的差异。 t。在 analagous 操作-值情况 (这是控制算法最重要的情况) 中残差梯度算法可以精确地简化为原始算法。

鲁棒性，适用于线性和非线性函数逼近器。在线性情况下，收敛总是对唯一的 w 最小。但是，残差梯度法的收敛至少有三种方法是不能令人满意的。第一个是，从经验上讲，它比半梯度法慢得多。实际上，这种方法的支持者们已经提出通过将它与更快的半梯度方法相结合来提高它的速度，然后逐渐切换到残差梯度以保证收敛 (Baird 和 Moore, 1999)。第二种不满意残差梯度算法的方法是它似乎仍然收敛于错误的值。它在所有的表格情况下都得到了正确的值，例如 A-split 示例，对于这些 Bellman 的精确解来说也是如此

方程是可能的。但是如果我们用真实的函数逼近来检验例子，那么残差-梯度算法，实际上是客观的，似乎找到了错误的值函数。最能说明问题的例子之一是在 A-split 示例上的变化，该示例称为 A-presplit 示例，如前一页所示，其中 residual-gradient 算法找到与原始版本相同的糟糕解决方案。这个例子直观地表明最小化 BE (residual-gradient 算法肯定会这么做) 可能不是一个理想的目标。第三部分解释了残差梯度算法收敛性不佳的原因。就像第二种方法一样，第三种方法也是一种客观的问题，而不是用任何特定的算法来实现它。

11.5 Bellman 错误是不可学的

我们在这一节介绍的学习能力的概念与机器学习中常用的概念不同。在这里，假设是“可学的”，如果它是有效的可学的，这意味着它可以在一个多项式中学习，而不是一个指数的例子。在这里，我们用一种更基本的方式来使用这个术语，指的是在任何情况下都可以学习。事实证明，强化学习中许多明显的兴趣是无法从大量的经验数据中习得的。这些量定义得很好，可以根据环境内部结构进行计算，但不能根据所观察到的特征向量、行为和奖励序列进行计算或估计。我们说他们是学不会的。最后两节介绍的 Bellman 错误目标 (BE) 在这种意义上是无法学习的。从可观测数据中无法得知 Bellman 错误目标，这可能是不去寻找它的最大原因。为了让学习的概念变得清晰，让我们从一些简单的例子开始。考虑下面的两个 Markov 奖励过程 3 (MRPs):

当两条边离开一个状态时，假设两个跃迁发生的概率都是相等的，这些数字表示所收到的奖励。所有的状态看起来都一样; 它们都产生相同的单分量特征向量 x = 1，并且都近似于 w。因此，数据轨迹中唯一变化的部分就是奖励序列。左边的 MRP 保持相同的状态，随机释放出无穷无尽的 0 和 2，每一个都有 0.5 的概率。正确的 MRP，在每一步上，要么停留在它的当前状态，要么。

如果观察到状态序列，而不是只观察到状态序列，它们当然会被估计相应的特征向量。所有 mrp 都可以被认为是 MDPs，在所有的状态下都有一个动作; 我们对 MRPs 的结论同样适用于 MDPs。

切换到另一个，概率相等。奖励是确定性的 MRP, 总是从一个状态 0 和 2 的, 但是因为每个州同样可能在每一步, 可观测的数据又是无穷无尽的 0 和 2 s 随机, 与由左 MRP。(我们可以假设正确的 MRP 以两种状态之一随机启动，概率相等。) 因此，即使给定无限的数据，也不可能知道这两个 mrp 中是哪一个生成的。特别是，我们无法判断 MRP 是否有一两个状态，是随机的还是确定性的。这些东西是学不会的。这一对 mrp 还说明 VE 目标 (9.1) 是不可学习的。如果 = 0, 那么的真实值三个州 (在两个可机读护照), 从左到右, 是 1 0,2。假设 w = 1。那么左 MRP 为 0，右 MRP 为 1。由于 VE 在这两个问题中是不同的，但是生成的数据具有相同的分布，因此无法了解 VE。VE 不是数据分布的唯一函数。如果它是学不到的，那么“VE”怎么可能成为学习的目标呢? 如果一个目标不能被学习，它确实会把它的效用变成问题。然而，在 VE 的情况下，有一条出路。注意, 相同的解决方案,w = 1, 都是最优的可机读护照上面 (假设 是相同的两个州中区分正确的 MRP)。这是巧合吗? 还是说所有数据分布相同的 MDPs 都有相同的最优参数向量? 如果这是真的——我们接下来将证明它是真的——那么 VE 仍然是一个可用的目标。VE 不是可学的，但是优化它的参数是! 要理解这一点，引入另一个自然目标函数是很有用的，这一次是一个明显可以学习的函数。一个总是可以观察到的错误是，在每个时间的值估计值和那个时间的收益之间。均方返回错误, 表示再保险公司的期望, 在 , 广场上的错误。在 on-policy 的情况下，可以写入 RE

因此，除了不依赖于参数向量的方差项之外，这两个目标是相同的。两个目标必须具有相同的最优参数值 w∗。总体关系在图 11.4 的左侧进行了总结。∗ 锻炼 11.4 证明 (11.24)。提示: 写再保险作为期望的可能状态年代期望平方误差的考虑到圣 = s。然后加减的真正价值的年代错误 (平方之前), 分组的真实价值减去回来估计价值增加的真正价值。然后，如果你展开平方，最复杂的项最后会是 0，剩下 (11.24)? 现在让我们回到现实。BE 就像 VE，它可以从 MDP 的知识中计算出来，但不能从数据中学习。但它不像 VE，它的最小解是不可学的。下一页的方框给出了一个反例——两个 mrp 生成相同的数据分布，但其最小参数向量不同，证明了最优参数向量不是

图 11.4: 数据分布、MDPs 和各种目标之间的因果关系。左，蒙特卡罗目标: 两个不同的 MDPs 可以产生相同的数据分布然而，也产生了不同的 VEs，证明 VE 的目标无法从数据中确定并不是学得来的。然而, 所有这些类型必须具有相同的最优参数向量,w∗! 此外, 同样的 w∗ 可以从另一个目标, 确定再保险, 是独一无二的由数据分布确定。因此 w∗ 类型和再保险可学的虽然不是。正确，引导目标: 两个不同的 MDPs 可以生成相同的数据分布也产生不同的

微分方程，且具有不同的最小参数向量; 这些在数据分布中是无法学习的。PBE 和 TDE 目标及其 (不同的) 最小值可以直接从数据中确定，因此是可以学习的。

因此不能从数据中学习。我们考虑的另一个引导目标，PBE 和 TDE，可以从数据 (是可学习的) 中确定，并确定通常不同的最优解和最小值。一般情况总结在图 11.4 的右边。因此，BE 是不可学习的; 它不能从特征向量和其他可观测数据来估计。这将 BE 限制为基于模型的设置。没有一种算法可以在不访问特征向量之外的底层 MDP 状态的情况下最小化 be。residual-gradient 算法只能最小化 BE，因为它允许从相同状态重复采样——不是具有相同特征向量的状态，而是保证为相同底层状态的状态。我们现在可以看到，这是不可能的。最小化 BE 需要访问名义上的、底层的 MDP。这是在第 273 页上的 A-presplit 示例中所标识的范围之外的一个重要限制。所有这些都将更多的注意力转向 PBE。

11.6 Gradient-TD 方法

我们现在考虑 SGD 方法来最小化 PBE。作为真正的 SGD 方法，这些梯度- td 方法在非策略训练和非线性函数逼近下具有鲁棒收敛性。记住，在线性情况下总会有一个精确的解，TD 不动点 wTD，在这个点 PBE 为 0。这个解可以通过最小二乘方法 (第 9.8 节) 找到，但只能通过参数数的二次复杂度 O(d2) 的方法找到。我们转而寻找一种 SGD 方法，它应该是 O(d) 并且具有鲁棒收敛性。梯度- td 方法接近于实现这些目标，代价是计算复杂度的粗略增加一倍。为了得到 PBE 的 SGD 方法 (假设线性函数逼近)，我们首先将目标 (11.22) 展开并重写为矩阵形式:

= 吗?wΠ 吗?DΠw = 吗?wDX

X ? DX

−1 x ? Dw (11.25) (使用 (11.14) 和身份 Π?X DΠ= DX ? ?DX−1 X ? D) =

X ? Dw X X ? ? ? DX−1 ? ? Dw。(11.26) 关于 w 的梯度是。

PBE(w)= 2 ? X ?Dw X ? DX−1 ? X ? Dw。要将它转化为 SGD 方法，我们必须在每一个时间步上对具有这个量作为期望值的东西进行采样。让我们以 的分布状态下访问行为的政策。以上三个因素都可以用这个分布下的期望来表示。例如，最后一个因素可以写出来

X ?Dw =

s (s)x(s)w(s)= E[t txt],

也就是半梯度 TD(0) 更新 (11.2) 的期望。第一个因素是这个更新的梯度的转置:

E(t txt)?= E ? t ?t x ?t = E

t (Rt + 1 + w 吗?xt + 1−w ? xt)? x ?t (使用情景 t) = E

x t(xt + 1−xt)?t

.

最后，中间因子是特征向量的期望外积矩阵的逆:

X ?DX =

s (s) 用于?s = E

xtx

t

把这些期望代入我们表达式中的三个因子，得到 PBE 的梯度 PBE(w)= 2 e ?x t(xt + 1−xt)?t E xtx ?t−1 E[t txt]。(11.27) 用这种形式来写渐变，可能并没有明显的进展。它是三个表达式的乘积，第一个和最后一个不是独立的。它们都依赖于下一个特征向量 xt+1; 我们不能简单地对这两个期望进行抽样，然后将样本相乘。这将给我们一个有偏见的梯度估计，就像在原始的残差梯度算法。另一种方法是分别估计这三个期望，然后把它们结合起来，得出对梯度的无偏估计。这将工作, 但需要大量的计算资源, 特别是存储前两个预期,d×d 矩阵, 并计算的倒数第二。这个想法可以改进。如果对三个期望中的两个进行估计和存储，那么第三个期望就可以与两个存储的量一起采样和使用。例如，您可以存储第二两个量的估计值 (使

用 9.8 节中的增量反更新技术)，然后对第一个表达式进行示例。不幸的是，总体算法仍然是二次复杂度 (O(d2))。将一些估计值单独存储，然后将它们与示例组合在一起的想法是一个很好的想法，并且也用于 Gradient-TD 方法。Gradient-TD 方法在 (11.27) 中估计和存储第二个因子的乘积。这些因素是 d×d 矩阵和一维矢量, 所以他们的产品只是一个维向量, 像 w 本身。我们用 v 表示第二个学习向量:

E v
xtx
t
−1 E[t txt]。(11.28) 这种形式对线性监督学习的学生很熟悉。线性最小二乘问题的解决方案, 试图近似 t t 的特性。标准的 SGD 方法，用于增量地寻找向量 v，从而最小化预期的平方。错误

v ?xt− t t 2 被称为最小均方 (LMS) 规则 (此处增广) 具有重要抽样比率):

vt + 1 。= vt + t
t−v ?t xt
xt,

其中 > 0 是另一个步长参数。我们可以使用这种方法，通过 O(d) 存储和每步计算有效地实现 (11.28)。给定一个已存储的估计 vt 近似 (11.28)，我们可以使用基于 (11.27) 的 SGD 方法更新我们的主要参数向量 wt。最简单的规则是

wt + 1 =− 那样 1 2 PBE(wt) (一般 SGD 规则)
= − 1 2 2E
x t(xt + 1−xt)?t
和
xtx
t
−1 E[t txt](从 (11.27)) = wt + E
x t(xt− xt + 1)?t
和
xtx
t
−1 E(t txt)(11.29) wt + E
x t(xt− xt + 1)?t
vt (基于 (11.28)) wt + t(xt− xt + 1)x ?t vt。(抽样)

280 年第 11 章: 带有近似的政策外方法

这个算法叫做 GTD2。注意，如果最终的内积 (x? 首先完成 t (vt)，然后整个算法的复杂度为 O(d)。在替换 vt 之前，再做一些分析步骤，就可以得到稍微更好的算法。

wt + 1 = wt + E
x t(xt− xt + 1)?t
和
xtx
t
−1 E[t txt] = wt +
和
txtx 吗?t
− E
txt x + 1 ?t
和
xtx
t

−1 E[t txt] = wt +
和
xtx
t
− E
txt x + 1 ?t
和
xtx
t
−1 E[t txt] = wt +
E[xt t t]− E ? txt x + 1 ?t E xtx ?t−1 E[t txt]
wt +
E[xt t t]− E ? txt x + 1 ?t vt(基于 (11.28)) wt + t
txt− xt x + 1 ?t vt
, (抽样)

也就是 O(d) 如果最终产物是 x?t (vt) 是先做的。该算法被称为带有梯度修正 (TDC) 的 TD(0) 或作为 GTD(0)。图 11.5 显示了一个示例和 Baird 反例上 TDC 的预期行为。如预期的那样，PBE 降至零，但请注意参数向量的各个分量不会接近零。事实上，这些价值还远远没有达到
5
2
0
-2.34
0 1000 步
0 1000 年清洁工

图 11.5:TDC 算法对 Baird 反例的行为。左边的是显示典型的单次运行，右边显示的是该算法的预期行为更新是同步完成的 (类似于 (11.9)，除了两个 TDC 参数向量)。步大小 = 0.005, = 0.05。

一个最优解,v̂(s)= 0, 所有的年代, 而 w 必须成正比 (1,1,1,1,1,1,4,−2) 吗?。在 1000 次迭代之后，我们仍然远离一个最优解，正如我们从 VE 中看到的，它仍然几乎是 2。系统实际上正在收敛到一个最优的解决方案，但进展极其缓慢，因为 PBE 已经如此接近于零。GTD2 和 TDC 都涉及两个学习过程，一个是 w 的初级学习过程，一个是 v 的二级学习过程。初级学习过程的逻辑依赖于二级学习过程的完成，至少是近似的，而二级学习过程则不受第一级学习的影响。我们称这种不对称依赖为级联。在叶栅中，我们经常假设二次学习过程进行得更快，因此总是在其渐近值，随时准备和准确地帮助初级学习过程。这些方法的收敛证明经常明确地做出这种假设。这些被称为双时间尺度证明。快速时间尺度是中等学习过程的时间尺度，较慢的时间尺度是初级学习过程的时间尺度。如果 的步长是主要学习过程, 和 的步长是次要的学习过程, 那么这些收敛性证明通常会要求在极限情况下 0 和 →→0。梯度- td 方法目前是最能被理解和广泛使用的稳定的非政策方法。有行动的扩展值和控制 (《GQ》,Maei et al .,2010), 合格的痕迹 (GTD( ) 和《GQ》( )Maei,2011; 和非线性函数逼近 (Maei 等，2009)。也有人提出了介于半梯度 TD 和梯度 TD 之间的混合算法 (Hackman, 2012; 白色, 白色,2016)。混合- td 算法在目标和行为策略非常不同的状态下表现得像梯度- td 算法，在目标和行为策略相同的状态下表现得像半梯度算法。最后，将 Gradient-TD idea 与 proximal method 和 control variates 相结合，生成更高效的方法 (Mahadevan et al.，2014; 杜 et al .,2017)。

11.7 Emphatic-TD 方法

现在我们转向第二种主要策略，它已经被广泛地探索，以获得一种具有函数逼近的廉价而有效的脱机学习方法。回想一下, 线性 semi-gradient TD 方法是有效和稳定的训练在政策下分布时, 我们在 9.4 节, 这与积极的矩阵 A 的明确性 (9.11)4 和政策之间的匹配状态分布 和状态转换概率 p(| 年代,) 目标的政策。在 off-policy 学习中，我们使用重要性抽样对状态转换进行重新加权，以便它们适合于学习目标策略，但是状态分布仍然是行为策略的分布。有一个不匹配。一个自然的想法是，以某种方式重新调整状态，强调某些状态，而不强调其他状态，以便将更新的分布返回到相应的策略分布。届时将会有一场比赛，而现有的结果将会带来稳定和融合。这就是。

4 off-policy 情况下, 矩阵通常定义为 Es b

x(s)E ? x(圣 + 1)? 吗? 圣 = s, ? ?。

着重介绍了在 9.11 事件中进行政策培训的方法。实际上，“政策上的分配”的概念并不十分正确，因为有许多政策上的分配，其中任何一个都足以保证稳定。考虑一个不打折扣的情景问题。情节结束的方式完全取决于过渡的可能性，但是情节开始的方式可能有几种不同。然而，如果所有的状态转换都是由于目标策略而导致的，那么结果的状态分布就是一个 on-policy 分布。在结束之前，您可能会开始接近终点站状态，并且只访问少数几个有高概率的状态。或者你可能从很远的地方出发，在结束之前经过许多州。两者都是政策上的分布，用线性半梯度法对两者进行训练将保证是稳定的。无论过程如何启动，只要遇到的所有状态更新到终止，就会产生策略上的分发结果。如果有折现，可以将其视为部分或概率终止。如果 = 0.9, 那么我们可以考虑用概率 0.1 进程终止在每个时间步, 然后立即重新启动的状态转变。贴现的问题是不断地终止和重启的概率 1− 在每一步。这种考虑折现的方法是一个更普遍的伪终止的例子，它不影响状态转换的序列，但确实影响学习过程和学习的量。这种伪终止对于脱机学习很重要，因为重新启动是可选的——记住我们可以以任何方式启动——而终止免除了将遇到的状态包含在策略分布中的需要。也就是说，如果我们不把新州视为重新开始，那么贴现很快就会给我们一个有限的政策分配。一步 Emphatic-TD 算法学习情景状态值被定义为: t = Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt),

wt + 1 = wt + Mt t t v̂(St,wt), 太 = t−1 太 −1 +, , 利息, 是任意的, 太强调, 被初始化为太 −1 = 0。这个算法对 Baird 的反例如何执行? 图 11.6 显示了参数向量分量的期望轨迹 (对于所有 t 都等于 1 的情况)，有一些振荡，但最终所有的都收敛，VE 趋于 0。这些轨迹是通过迭代计算参数向量轨迹的期望而得到的，而不是由于转换和奖励的抽样而产生的任何方差。由于该算法对 Baird 反例的方差很大，在计算实验中几乎不可能得到一致的结果，所以我们没有直接给出应用该算法的结果。该算法在理论上收敛于最优解，但在实际应用中并不收敛。下一节我们将讨论如何减少所有这些算法的方差。

11.9。减少方差 283 年

图 11.6: 对 Baird 的反例的期望的单步强化- td 算法的行为。步长是 = 0.03。

11.9 减少方差

政策外学习本质上比政策内学习具有更大的差异。这并不奇怪; 如果您接收到的数据与策略的关系不太密切，那么您应该对策略的价值了解得更少。在极端情况下，一个人可能什么也学不到。例如，你不能指望通过做饭来学习开车。只有当目标和行为政策是相关的，如果他们访问相似的国家并采取类似的行动，一个人才能在政策之外的培训中取得显著的进展。另一方面，任何政策都有许多邻国，许多相似的政策在访问过的国家和选择的行动中有相当大的重叠，但它们并不相同。雷森 d 本部ˆoff-policy 学习是使概括的混乱关系大量 related-but-not-identical 政策。问题仍然是如何充分利用这段经历。既然我们已经有了一些在期望值上是稳定的方法 (如果步骤大小设置正确的话)，那么注意力自然就会转向减少估计的方差。有很多可能的想法，我们可以在这篇介绍性文章中讨论其中的一些。为什么在基于重要性抽样的非策略方法中控制方差特别重要? 正如我们所看到的，重要性抽样通常涉及政策比率的产品。比率总是期望为 1(5.13)，但它们的实际值可能非常高或低至 0。连续比率是

不相关的，所以它们的产品在期望值上也是一样的，但是它们的方差可能非常大。回想一下，在 SGD 方法中，这些比率将步长相乘，因此高方差意味着采取在大小上差异很大的步骤。这对 SGD 是有问题的，因为偶尔会有非常大的步骤。它们不能大到把参数带到空间的一个有非常不同的梯度的部分。SGD 方法依赖于对多个步骤进行平均，以获得对梯度的良好感觉，如果它们从单个样本中进行较大的移动，就会变得不可靠。

如果步长参数设置得足够小以防止这种情况发生，那么预期的步骤可能会非常小，从而导致非常慢的学习。动量的概念 (Derthick, 1984)，Polyak- ruppert 平均值 (Polyak, 1990;Ruppert,1988;Polyak 和 Juditsky, 1992)，或者进一步扩展这些思想可能会有很大的帮助。为参数向量的不同分量自适应地设置独立步长的方法也是相关的 (例如，Jacobs, 1988;Sutton, 1992b, c)，以及 Karampatziakis 和 Langford(2010) 的“重要性体重意识”更新。在第 5 章中，我们看到了加权重要性抽样比普通重要性抽样更能表现得更好，方差更新更低。然而，将加权重要性抽样应用于函数逼近是一种挑战，而且可能只能用 O(d) 复杂度完成 (Mahmood 和 Sutton, 2015)。树备份算法 (第 7.5 节) 表明，在不使用重要抽样的情况下，可以执行一些策略外的学习。这一想法已经扩展到 off-policy 案例，由 Munos、Stepleton、Harutyunyan 和 Bellemare(2016) 以及 Mahmood、Yu 和 Sutton(2017) 提出稳定和更有效的方法。另一种补充策略是，允许目标策略在一定程度上由行为策略决定，其方式是，它与行为策略之间永远不会有如此大的差异，以创建重要的抽样比率。例如，目标策略可以通过引用行为策略来定义，如 Precup 等人 (2006) 提出的“识别器”。

11.8 总结

离线学习是一个很有诱惑力的挑战，考验我们在设计稳定有效的学习算法方面的独创性。表列 Q-learning 使偏离策略的学习看起来很容易，它对预期 Sarsa 和树备份算法有自然的归纳。但是正如我们在这一章中看到的，将这些思想推广到重要的函数逼近，甚至是线性函数逼近，会带来新的挑战，迫使我们加深对强化学习算法的理解。为什么要走这么长的路? 寻求偏离策略的算法的一个原因是在处理探索和开发之间的权衡时提供灵活性。另一种方法是将行为从学习中解放出来，避免目标政策的专制。TD 学习似乎提供了同时学习多个事物的可能性，即使用一种经验流同时解决多个任务。我们当然可以在特殊情况下这样做，只是不是在我们想要或想要的任何情况下。在本章中，我们将政策外学习的挑战分为两部分。第一部分，纠正行为策略的学习目标，直接处理使用前面为表格案例设计的技术，尽管其代价是增加更新的方差，从而减缓学习。对于偏离策略的学习来说，高方差可能始终是一个挑战。脱机学习的挑战的第二部分是半梯度 TD 方法的不稳定性，包括自举。我们寻求强大的功能

近似值，非政策学习，以及 bootstrapping TD 方法的效率和灵活性，但在一种算法中结合这三种致命的三元算法，并没有引入不稳定性的可能性，这是一种挑战。有几次尝试。最流行的方法是在 Bellman 错误 (即 Bellman 残余物) 中寻求执行真正的随机梯度下降 (SGD)。然而，我们的分析得出的结论是，在很多情况下，这并不是一个吸引人的目标，而且无论如何，用学习算法是不可能实现的。另一种方法，Gradient-TD 方法，在预计 Bellman 错误中执行 SGD。在复杂度为 O(d) 的情况下，PBE 的梯度是可以学习的，但是代价是需要第二个参数向量，并且需要第二个步骤大小。最新的方法系列，强调- td 方法，改进了一个旧的更新权重，强调了一些，而不强调其他的。通过这种方式，它们恢复了使用计算简单的半梯度方法使策略学习稳定的特殊属性。非政策学习的整个领域相对较新，也不稳定。哪些方法是最好的，甚至是适当的，目前尚不清楚。本章末尾介绍的新方法的复杂性真的有必要吗? 其中哪一种可以有效地与方差约简方法结合? 偏离政策学习的可能性仍然诱人，实现它的最佳方式仍然是个谜。

11.9 书目的和历史的言论

11.1 第一个 semi-gradient 方法是线性 TD( )(萨顿,1988)。这个名字“半梯度”是最近才出现的 (Sutton, 2015a)。在 Sutton、Mahmood 和 White(2016) 之前，一般输入-采样率的半梯度非政策 TD(0) 可能没有明确的说明，但是 Precup、Sutton 和 Singh(2000) 引入了动作价值形式，他们也采用了这些算法的合格跟踪形式 (见第 12 章)。他们持续的、未打折的形式还没有得到充分的探索。这里给出的 n 步形式是新的。

11.2 最早的 wto -2w 示例是 Tsitsiklis 和 Van Roy(1996) 提出的还在 263 页的方框中介绍了具体的反例。Baird 的反例来自 Baird(1995)，尽管我们在这里展示的版本稍作修改。函数近似的平均方法是由 Gordon (1995, 1996b) 开发的。其他的不稳定性的例子，还有非政策的 DP 方法和更复杂的函数近似方法，由 Boyan 和 Moore(1995) 给出。Bradtke(1993) 给出了一个在线性二次调节问题中使用线性函数逼近的 Q-learning 收敛于不稳定策略的例子。

11.3 死亡三位一体首先由 Sutton(1995 年 b) 鉴定并彻底分析作者 Tsitsiklis 和 Van Roy(1997)。“致命三合会”这个名字是由 Sutton (2015a) 起的。11.4 这种线性分析是 tsitsitsiklis 和 Van Roy(1996) 首创的;1997 年),

包括动态编程操作符。像图 11.3 这样的图表是由 Lagoudakis 和 Parr(2003) 介绍的。我们称为传达员操作符, 表示 B , 更常见的 T 来表示, 称为“动态规划算子, 而广义的形式, 表示 T( ), 被称为“TD( ) 操作符”(Tsitsiklis 和 Van 罗伊,1996,1997)。

11.5 BE 首先作为动态规划的目标函数被提出施韦策和 Seidmann(1985)。Baird(1995, 1999) 将其扩展为基于随机梯度下降的 TD 学习。在文献中，BE 最小化常被称为 Bellman 剩余最小化。最早的 a 分裂的例子是 Dayan(1992)。这里给出的两种形式是 Sutton 等人 (2009a) 介绍的。

11.6 本节的内容对本文来说是全新的。

11.7 Gradient-TD 方法引入 Sutton Szepesvári, 和 Maei(2009 b)。本节重点介绍的方法由 Sutton 等人 (2009a) 和 Mahmood 等人 (2014) 介绍。由 Mahadeval 等人 (2014) 开发了近端 TD 方法的一个主要扩展。到目前为止，Gradient-TD 及其相关方法最敏感的实证研究由 Geist 和 Scherrer(2014)、Dann、Neumann 和 Peters(2014)、White(2015) 和 Ghiassian、White、White、White 和 Sutton(准备中) 进行。Yu(2017) 介绍了梯度- td 方法理论的最新进展。

11.8 Sutton、Mahmood 和 White(2016) 着重介绍了 c- td 方法。Yu(2015 年) 建立了完整的收敛证明和其他理论;2016;Yu, Mahmood, and Sutton, 2017)，Hallak, Tamar, Mannor(2015)，以及 Hallak, Tamar, Munos, Mannor(2016)。

12. 第十二章资格的痕迹

资格追索是强化学习的基本机制之一，在受欢迎的 *TD(*) 算法，是指合格的使用痕迹。几乎任何时间差异 *(TD)* 方法，如 *Q-learning* 或 *Sarsa*，都可以与合格跟踪相结合，以获得更通用的方法，从而可以更有效地学习。资格跟踪统一和推广 *TD* 和蒙特卡罗方法。*TD* 方法增强与资格的痕迹时, 产生一个家庭的方法生成一个频谱一端蒙特卡罗方法 *(= 1)* 和一步 *TD* 方法另 *(= 0)*。中间是中间方法，它们通常比两个极端方法都要好。资格跟踪还提供了一种在线实现蒙特卡罗方法的方法，以及在没有发作的连续问题上。当然，我们已经看到了一种统一 *TD* 和蒙特卡罗方法的方法: 第 *7* 章的 n 步 *TD* 方法。除了这些之外，合格跟踪还提供了一个优雅的算法机制，具有重要的计算优势。短期记忆的机制是一个向量, 资格跟踪 zt 型 Rd, 相似之处长期体重向量 wt Rd。粗略的想法是, 当一个组件 wt 参与生产的估计价值, 那么相应的组件 zt 型骤然上升, 然后开始消失。如果在跟踪返回到 *0* 之前出现了一个非零的 *TD* 错误，那么就会在 wt 的那个组件中进行学习。痕迹减退参数 *[0,1]* 决定跟踪的速度下降。与 n 步方法相比，资格跟踪的主要计算优势是只需要一个跟踪向量，而不是最后 n 个特征向量的存储。学习也会不断地、均匀地发生，而不是被延迟，然后在这一集的结尾出现。此外，学习可以在遇到状态后立即发生并影响行为，而不是被延迟 n 步。合格跟踪说明，学习算法有时可以用不同的方式实现，以获得计算优势。许多算法最自然地被表述为状态值的更新，它基于在未来多个时间步骤中跟随该状态的事件。例如，蒙特卡罗方法 *(*第 *5* 章*)* 基于所有未来奖励更新状态，n 步 *TD* 方法 *(*第 *7* 章*)*

更新基于未来 n 个奖励和状态 n 步。这种基于对更新状态的展望的公式称为前瞻性视图。*Forward* 视图的实现总是有些复杂，因为更新依赖于以后的东西，而这些东西当时是不可用的。然而，正如我们在本章中所展示的，使用一个使用当前 *TD* 错误的算法来实现几乎相同的更新 *(*有时甚至是完全相同的更新*)*，使用一个合格跟踪来回顾最近访问的状态。这些观察和实现学习算法的替代方法称为向后视图。后向视图、前向视图和后向视图之间的转换，以及它们之间的相等性，可以追溯到时间差异学习的引入，但自 *2014* 年以来变得更加强大和复杂。这里我们介绍现代观点的基础。和往常一样，我们首先充分发展状态值和预测的思想，然后将它们扩展到动作值和控制。我们首先针对政策上的情况开发它们，然后将它们扩展到政策外的学习。我们的处理特别注意线性函数逼近的情况，其结果具有更强的适用性。所有这些结果也适用于表列和状态聚集情况，因为这些是线性函数逼近的特殊情况。

12.1 的 -return

在第 7 章中，我们定义了 n 步返回作为前 n 个奖励的总和加上 n 个步骤中所达到的状态的估计值，每一个都适当地折现 (7.1)。对于任何参数化函数近似器，这个方程的一般形式是 Gt:t + n。= Rt + 1 + Rt + 2 + ••• + n−1 Rt + n + nv̂(wt 圣 + n + n−1),0 t t−n、v(12.1)(s,w)

的近似值 s 给定的权向量 w(第九章), 和 t 是一集的时间终止, 如果任何。我们在第 7 章指出, 每个 n-step 回报, 对 n 1, 是一个有效的更新表格学习更新的目标, 就像它是一个近似 SGD 学习更新, 如 (9.7)。现在我们注意到，一个有效的更新不仅可以针对任何 n 步返回，还可以针对不同 ns 的任何 n 步返回的平均值。例如，可以对目标进行更新，目标是两步返回的一半和四步返回的一半: 1 2 gt:t + 2 + 1 2 gt:t + 4。任何 n 步返回的集合都可以这样平均，即使是无穷大设置，只要组件返回的权重为正，sum 为 1。复合返回具有与单个 n 步返回相似的错误减少属性 (7.3)，因此可以使用有保证的收敛属性构造更新。平均产生了大量新的算法。例如，一个人可以平均一步和无限步返回来获得另一种与 TD 和蒙特卡罗方法相关的方法。原则上，我们甚至可以使用 DP 更新来平均基于经验的更新，以获得基于经验和基于模型的方法的简单组合 (参见第 8 章)。平均简单组件更新的更新称为复合更新。复合更新的备份关系图由每个组件更新的备份关系图组成，每个组件更新在其上面有一条水平线，下面是权重部分。

1 2 例如，在本节开始时提到的案例的复合更新，混合了两步返回的一半和四步返回的一半，将图显示在右边。复合更新只能在组件更新时间最长的时候完成。例如，右边的更新只能在 t+4 时刻完成 t+4 时刻形成的估计值。一般来说，我们希望限制最长组件更新的长度，因为更新中相应的延迟。TD( ) 算法可以被理解为一个特定的方式平均 n-step 更新。这种平均包含所有 n-step 更新, 每个加权比例 n−1([0,1]), 和规范化的因素 1− , 确保权重之和为 1(图 12.1)。对回归结果更新, 称为 -return, 定义基于状态的形式

G t 。=(1−) ∞ 吗?n = 1 n−1 gt:t + n。(12.2)

图 12.2 进一步说明了加权 -return n-step 返回的顺序。一步返回给定的最大重量,1− ; 两步返回都将最大重量 (1−) ; 三步返回给出的重量 (1−) 2; 等等。权重 褪色的每个额外的步骤。在达到一个终端状态后，所有后续的 n 阶返回都等于常规返回 Gt. If

TD( )

(1−) 2

t T−−1 −1

圣 RT

= 1

图 12.1: 备份双字母组合 TD( )。如果 = 0, 那么整个更新减少了它的第一个组件, 一步 TD 更新, 而如果 = 1, 然后整体更新减少其最后一个组件, 蒙特卡洛更新。

图 12.2: 加权给出 -return 每个 n-step 回报。

我们想要，我们可以将这些终止后的条件与主要的和，产生分离。

G t =(1−) T−−1 ?n = 1 n−1 gt:t + n + T t−−1 gt, (12.3)

如图所示。这个方程使它清晰当 = 1。在这种情况下，主要的和为零，剩余的项减少到常规的回报。因此, 对于 = 1, 更新根据 -return 蒙特卡罗算法。另一方面, 如果 = 0, 然后 -return 减少 Gt:t + 1, 一步返回。因此, 对于 = 0, 更新根据 -return 是一个一步 TD 方法。练习 12.1 返回可以递归地写在第一个奖励和本身一步后 (3.9), 所以可以 -return。导出类似的递归关系从 (12.2) 和 (12.1)。? 12.2 运动参数 特征速度指数加权图 12.2 脱落, 因此多远未来 -return 算法看起来在决定它的更新。但利率因素如 有时是一个尴尬的方式描述的速度衰减。出于某些目的，最好指定一个时间常数或半衰期。 有关的方程和半衰期, , 加权序列的时间降至初始值的一半吗??

现在, 我们可以定义我们的第一个学习算法基于 -return: 离线 -return 算法。作为一种离线算法，它不会改变集内的权重向量。的最后一集, 整个序列的离线更新是根据我们平时 semi-gradient 规则, 使用 -return 作为目标:

wt + 1 . = wt +

wt G t−v̂(圣)

wt v̂(圣),t = 0,。T−1。(12.4)

-return 给我们的另一种方式之间移动顺利蒙特卡罗和一步 TD 方法可以与 n-step 引导方式发达在第 7 章。在那里，我们评估了 19 个状态随机行走任务的有效性 (例子 7.1，第 144 页)。图 12.3 显示了离线 -return 算法的性能在这个任务,n-step 方法 (重复从图 7.2)。前面描

述的实验一样, 除了 -return 算法我们不同的 代替 n。使用的性能测量是正确估计之间的均方根误差和估计的每个状态测量值的最后一集, 平均在第十集和 19 个州。注意, 离线 -return 算法的总体性能是可比的 n-step 算法。在这两种情况下我们得到最佳性能的中间值引导参数,n 为离线 -return n-step 方法和 的算法。

0.4 最后是 RMS 错误。这一集的第一集 0.35 10 集

0.3

0.25

0.4 - 0.2 0 0.8 - 0.6 1

0.4 - 0.2 0 0.8 - 0.6 1

图 12.3:19 个随机游走的结果 (例如 7.1): 离线 -return 的性能与 n 步 TD 方法并行的算法。在这两种情况下，中间值引导参数 (或 n) 表现最好。结果与离线 -return 算法更能最好的 和 的值, 和在高 。

到目前为止，我们一直采用的方法就是我们所说的学习算法的理论观点。对于每一个访问过的州，我们都期待着所有未来的奖励，并决定如何最好地将它们结合起来。我们可以想象自己在状态流中前进，从每个状态中期待确定更新，如图 12.4 所示。在展望并更新一个状态之后，我们继续下一个状态，再也不必使用前一个状态。另一方面，对未来的状态进行反复的观察和处理，每次都是从它们前面的每个有利点开始。

图 12.4: 正向视图。我们通过展望未来的奖励和状态来决定如何更新每个状态。

12.2 TD( )

TD( ) 是一种最古老和最广泛使用的强化学习算法。这是第一个将正式的关系展示在一个更有理论的前视图和一个使用资格跟踪的更加计算一致的后视图之间的算法。在这里, 我们将展示经验, 它接近离线 -return 算法在前一节中提出的。TD( ) 改善离线 -return 算法在三个方面。首先，它更新了每一集的每一步的重量矢量，而不仅仅是在最后，因此它的估计可能会更快更好。其次，它的计算在时间上是平均分布的，而不是在这集的结尾。第三，它可以应用于持续的问题而不仅仅是偶发的问题。在本节中, 我们讨论 TD 的 semi-gradient 版本 ( ) 函数近似。与资格跟踪是一个矢量函数近似 zt 型 Rd 与相同数量的组件作为权向量 wt。而权向量是一个长期记忆, 积累在系统的生命周期, 资格跟踪是一个短期记忆, 通常持续时间少于一集的长度。资格追踪协助学习过程; 它们唯一的结果就是它们影响了权重向量，然后权重向量决定了估计值。在 TD( ) 资格跟踪矢量集开始时初始化为零, 增加在每个时间步的梯度值, 然后逐渐消退 : z −1 。= 0, zt 型. = zt−1 + v̂(St,wt),0 t t, (12.5) 贴现率和 参数介绍了在前面的小节中, 我们从今以后称之为痕迹减退参数。资格跟踪跟踪哪些组件的权向量有贡献, 积极的还是消极的, 最近的状态估值, “最近” 的定义。(回想一下, 在线性函数近似, v̂(St,wt) 只是一个特征向量,xt, 在这种情况下, 资格跟踪矢量就是一笔过去, 褪色, 输入向量。) 这一痕迹被认为是用来表示权重向量的每个组成部分是否有资格进行学习

加强事件发生时的变化。我们所关注的增强事件是时刻一步的 TD 错误。对状态值预测的 TD 误差是。

t。= Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt)。(12.6) 在 TD( ), 更新权向量在每个步骤与标量 TD 错误和向量资格跟踪:

wt + 1 . = wt + tzt。(12.7)

图 12.5: 落后的或机械的 TD( )。每次更新都取决于当前的更新 TD 错误结合了过去事件的当前合格跟踪。

TD( ) 是面向落后。在每个时刻，我们查看当前的 TD 错误，并根据该状态在当时对当前资格跟踪的贡献大小，将其向后分配到每个先前状态。我们可以想象自己沿着状态流运行，计算 TD 错误，并将它们返回到先前访问的状态，如图 12.5 所示。当 TD 错误和跟踪一起出现

时，我们得到 (12.7) 给出的更新，更改过去状态的值，以便在将来再次发生。为了更好地理解落后的 TD( ), 考虑在不同的 值会发生什么。如果 = 0, 那么通过 (12.5) 跟踪 t 就是梯度值对应于圣因此 TD( ) 更新 (12.7) 减少一步 semi-gradient TD 更新在第 9 章 (在表格的情况下, 简单的道明规则 (6.2))。这就是为什么这个算法叫做 TD(0) 根据图 12.5,TD(0) 是只有当前状态之前的一个状态被 TD 错误改变的情况。 值的增大, 但仍然 < 1, 之前更多的状态改变, 但每多暂时遥远的状态变化少, 因为相应的资格跟踪比较小, 所显示的人物。我们说，早期的州对 TD 错误的信任更少。如果 = 1, 那么信贷给美国早些时候下跌只有 每一步。这是实现蒙特卡洛行为的正确方法。例如, 记住,TD 错误, t, 包括一个尚未完全的 Rt + 1。通过这个回 k 步骤需要打折, 像任何奖励返回, 由 k 正是下降资格跟踪实现的。如果 = 1 和 = 1, 那么资格痕迹不随时间衰减。在这种情况下，该方法的行为就像蒙特卡罗方法，用于不打折扣的、周期性的任务。如果 = 1, 该算法也被称为 TD(1)。TD(1) 是一种实现蒙特卡罗算法的方法，比前面提到的算法更通用，并且显著增加了它们的适用范围。早期的蒙特卡罗方法仅限于情景任务，而 TD(1) 也可以应用于贴现连续任务。此外，TD(1) 可以增量地在线执行。蒙特卡罗方法的一个缺点是，在一段情节结束之前，他们什么也学不到。例如，如果蒙特卡罗控制方法采取的行动产生的报酬非常低，但没有结束这一集，那么在这一集中，代理重复这一行动的倾向将不会减弱。另一方面，在线 TD(1) 从不完整的正在进行的事件中学习了 n 步的 TD 方法，其中 n 步一直到当前的步骤。如果在某一集中发生了异常好的或坏的事情，基于 TD(1) 的控制方法可以立即学习并改变他们在同一集中的行为。它是揭示重温 19 个随机漫步的例子 (例如 7.1) 看看 TD( ) 在离线 -return 近似算法。这两种算法的结果如图 12.6 所示。对于每一个 值, 如果选择 最优 (或更小), 那么这两个算法执行几乎相同。如果选择 比是最优的, 然而, 然后 -return 算法只是有点糟糕而 TD( ) 是更糟, 甚至可能是不稳定的。这不是灾难性的 TD( ) 在这个问题上, 因为这些更高的参数值并不是一个想要使用, 但对于其他的问题可以是一个明显的弱点。

12.3 n-step 截断 -return 方法

最后是 RMS 错误。这一集的前 10 集 0.45

0.35 - 0.4

0.3

0.25

0.4 - 0.2 0 0.8 - 0.6 1 295 年

0.4 - 0.2 0 0.8 - 0.6 1

图 12.6:19 个随机游走的结果 (例如 7.1):TD( ) 旁边的性能的离线 -return 算法。这两种算法在低的情况下几乎完全相同。(小于最优) 值, 但 TD( ) 更糟糕的是在高 值。

线性 TD( ) 已经证明了在政策趋同的情况下, 如果减少步长参数随着时间的推移, 根据通常的条件 (2.7)。正如 9.4 节中所讨论的, 融合不是重量最小误差向量, 但到附近的权向量, 取决于 。中提供的绑定解决方案质量, 部分 (9.14) 现在可以推广到申请任何 。对于持续打折的情况，

即渐近误差不超过 1——1 * 尽可能最小的错误。 趋于 1, 绑定方法误差最小 (松开 = 0)。然而在实践中, = 1 通常是最穷的选择, 将在后面说明, 如图 12.14。练习 12.3 一些洞察 TD( ) 如何密切近似离线 -return 算法可以被看到了后者的误差项 (12.4)(在括号中) 可以写成 TD 错误 (12.6) 的总和为单个固定 w。这显示,(6.6) 的模式后, 使用递归关系 -return 你获得的 12.1 运动。? 练习 12.4 利用前面的锻炼的结果表明, 如果重量更新一集在每一步计算但不实际使用改变权重 (w 保持固定的), 然后 TD 的总和 ( ) 的重量更新将是相同的离线 -return 之和算法的更新。?

离线 -return 算法是一个重要的理想, 但这是有限的效用, 因为它使用 -return(12.2), 这是未知的, 直到最后一集。在

持续的情况下, -return 在技术上不知道, 因为它取决于 n-step 返回任意大 n, 从而在任意遥远未来的回报。然而, 依赖变得弱 longer-delayed 奖励, 下降 延迟的每一步。因此，一个自然的近似值是在一些步骤之后截断序列。我们现有的 n 步回报的概念提供了一种自然的方法，在这种方法中，缺失的回报被估计值所取代。一般来说, 我们定义了截断 -return 时间 t, 给定数据只是一些后来地平线,h,

G t:h 。=(1−) h t−−1 ?n = 1 n−1 gt:t + n + h t−−1 gt:h,h 0 t < t(12.9)

与 -return 如果你把这个方程 (12.3), 很明显, 地平线 h 是扮演相同的角色以前由 T, 终止的时间。而在 -return 有剩余重量给传统的 Gt 的回报, 这是最长的可用 n-step 回报,Gt:h(图 12.2)。截断 -return 立即产生一系列 n-step -return 算法类似于 n-step 第七章的方法。在所有这些算法, 更新延迟了 n 步, 只考虑第一个 n 奖励, 但现在所有 k-step 返回包含 1 k n(而早期 n-step 算法只用 n-step 返回), 加权几何如图 12.2 所示。在州值的情况下, 这个家庭的算法称为截断 TD( ), 或运输大亨 ( )。复合备份图, 如图 12.7 所示, 类似于 TD( )(图 12.1), 除了最多最长的组件更新 n 步而不是总是所有的方法

n−1

图 12.7: 备份图截 TD( )。

一集结束。运输大亨 ( ) 被定义为 (cf(9.15)):

wt + n . = t + n−1 +

G t:t + n−v̂(St,wt + n−1)

v̂(St,wt + n−1),0 t < t。

这种算法可以有效地实现，这样每一步的计算就不会与 n 成比例 (当然内存也必须如此)。n-step TD 方法, 没有更新在第 n−1 每一集的时间步骤, 和 n−1 额外的更新在终止。有效的实现依赖于事实 k-step -return 可以写一样

G t:t + k = v̂+(圣,wt−1) t + k−1 ? 我 = t ( ) 我 −t 吗? 我, (12.10)

在哪里

?t 。= Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt−1)。

练习 12.5 在这本书中 (通常在练习中) 我们已经确定，如果值函数保持不变，返回可以写成 TD 错误的和。为什么 (12.10) 是另一个例子吗? 证明 (12.10)。?

12.4 重新更新: 在线 -return 算法

选择截断参数 n 截断 TD( ) 包括一个权衡。n 应该大, 这样方法密切接近离线 -return 算法, 但它也应该小, 这样可以更快的更新, 可以影响行为。我们能两者兼得吗? 是的，原则上我们可以，尽管代价是计算复杂度。其思想是，在每次收集新的数据增量时，都要返回并重新执行当前事件开始以来的所有更新。新的更新将比您以前所做的更好，因为现在它们可以考虑到 time step 的新数据。即更新总是朝着一个 n-step 截断 -return 目标, 但他们总是使用最新的地平线。在每一段情节中，你都可以使用稍微长一点的视野，得到稍微好的结果。记得截断 -return 定义 (12.9)

G t:h 。=(1−) h t−−1 ?n = 1 n−1 gt:t + n + h t−−1 gt:h。

如果计算复杂度不是问题，那么让我们来看看理想情况下如何使用这个目标。本集以时间 0 的估计值开始，使用前一集末尾的权值 w0。当数据视界扩展到第一步时，学习就开始了。在第 0 步给出的估计值的目标是，给定数据到 horizon 1，只能是单步返回 G0:1，其中包括 R1 和 boot。w0 v̂(S1)。注意, 这正是 G 0:1, 总和的第一项方程的退化为零。使用这个更新目标，我们构建 w1。然后，在将数据层推进到步骤 2 之后，我们要做什么? 我们有新的数据形式的 R2 和 S2, 以及新的 w1, 所以现在我们可以构建一个更好的更新目标 G 0:2 第一更新从 S0 以及更好的更新目标 G 1:2 第二更新从 S1。使用这些改进的目标，我们在 S1 和 S2 重新执行更新，从 w0 开始，生成 w2。现在我们把地平线推进到第 3 步，重复，一路返回来产生 3 个新的目标，重新开始从原来的 w0 到 w3 的所有更新，等等。每一次视界被推进，所有的

更新都从 w0 开始使用来自前一个视界的权重向量重新完成。这个概念算法涉及到在这一集的多次传递，在每一个视界上，每一个都产生不同的权重向量序列。为了清晰地描述它，我们必须区分在不同视野下计算的权重向量。让我们使用 wh t 表示权重用于生成序列中的值在时间 t 地平线 h。第一个权向量 wh 0 在每一个序列是继承自上一次 (所以他们是相同的 h), 最后权向量 wh h 每个序列定义了最终的权向量序列的算法。在最终的视界 h = T 时，我们得到最终的权值 wT T，它将被传递，形成下一集的初始权值。有了这些公约，前一段所述的前三个顺序可以明确地给出:

h = 1: w1 . = 0 +
G 0:1−v̂(S0 w1 0)
v̂(S0,w1 0),
h = 2: w2 1 . = w2 0 +
G 0:2−v̂(S0 w2 0)
v̂(S0 w2 0), w2 2 . = w2 1 +
G 1:2−v̂(S1 w2 1)
v̂(S1 w2 1),
h = 3: w3 1 . = w3 0 +
G 0:3−v̂(S0,w3 0)
v̂(S0,w3 0), w3 2 . = w3 1 +
G 1:3−v̂(S1,w3 1)
v̂(S1,w3 1), w3 3 . = w3 2 +
G 2:3−v̂(S2,w3 2)
v̂(S2,w3 2)。更新的一般形式是
wh t + 1 . = wh t +
G t:h−v̂(St,wh t)
v̂(圣,wh t),0 t < h t。此更新与 wt 一起。= t 定义在线 -return 算法。完全在线, 在线 -return 算法确定一个新的权向量 wt 每一步 t 在一集, 仅使用信息在时间 t, 其主要缺点是计算复杂, 经过集的一部分经历了到目前为止在每一步。注意, 它比离线 -return 严格更复杂的算法, 通过的所有步骤时终止, 但不做任何更新在一集。作为回报，在线算法可以被期望比离线算法表现得更好，不仅在事件进行更新时，而离线算法没有更新时，而且在事件结束时，因为

权向量用于引导 (G t:h) 有更多的信息更新。如果仔细查看图 12.8，就可以看到这种效果，图 12.8 比较了 19 状态随机漫步任务中的两种算法。

0.5
最后是 RMS 错误。这一集的前 10 集 0.45
0.35 - 0.4
0.3
0.25
0.4 - 0.2 0 0.8 - 0.6 1
0.4 - 0.2 0 0.8 - 0.6 1

图 12.8:19 个随机游走的结果 (例如 7.1): 在线和离线 -return 算法的性能。这里的性能度量是在事件结束时的 VE，这应该是离线算法最好的情况。尽管如此，在线算法的表现还是相当不错。相比之下, = 0 线是相同的两种方法。

12.5 真实在线 TD( )

在线 -return 算法只是提出了目前表现最好的 temporal-difference 算法。它是一个理想的在线 TD( ) 只有接近。了, 然而, 网上 -return 算法非常复杂。是否有一种方法可以将这种前视

图算法转化为使用合格跟踪的有效的向后视图算法? 事实证明, 确实有一个精确的计算的实现在线 -return 算法的线性函数近似。这个实现被称为真正的在线 TD( ) 算法, 因为它是“真实”的理想在线 -return 算法比 TD( ) 算法。真在线 TD 的推导 ( ) 现在这里有点太复杂 (见下一节和论文的附录 van Seijen et al .,2016) 但是它的策略很简单。权重向量的序列产生的在线 -return 算法可以排成一个三角形:

w0 0 w1 0 w 1 1 w2 0 w 2 1 w 2 - 2 w3 0 w 3 1 w 3 2 w 3 3

wT 0 w T 1 w T 2 w T ••• w T T

这个三角形的一行在每个时间步上产生。结果是对角线上的权向量，wt t，是唯一真正需要的。第一个，w0 0，是这一集的初始权向量，最后一个，wT，是最终的权向量，每一个权向量，在这个过程中，wT，在更新的 n 步返回中起作用。在最后的算法中，对角权向量被重新命名，没有上标 wt 。策略就是找到一种紧凑、高效的计算方法。每个 wt 都是从 1 开始的。如果这样做, 对于线性情况下的 v̂(s,w)= w ? x(s), 然后到达真正的在线 TD( ) 算法:

wt + 1 . = wt + tzt +

w ?w t xt−

t−1 xt (zt 型 −xt), 我们在哪里使用了速记 xt 。= x(圣), t 被定义为在 TD( )(12.6), 和 zt 型定义为

zt 型。= zt−1 +

1− z 吗?t−1 xt

xt。(12.11) 该算法已被证明产生完全相同的权重向量序列,wt,0 t t, 随着在线 -return 算法 (van Seijen et al . 2016 年)。因此，图 12.8 左边的随机行走任务的结果也是该任务的结果。然而，现在的算法要便宜得多。真正的内存需求在线 TD( ) 与常规 TD( ) 完全相同, 而每一步计算增长了约 50%(还有一个内积 eligibility-trace 更新)。总的来说, 每一步的计算复杂度是 O(d), 一样的 TD( )。在方框中给出了完整算法的伪代码。

资格跟踪 (12.11) 中使用真实的在线 TD( ) 被称为荷兰跟踪区别于跟踪 (12.5) 用于 TD( ), 这被称为一个积累痕迹。早期的工作通常使用第三种称为替换跟踪的跟踪，这种跟踪只定义为表格或者二进制特征向量，比如由 tile 编码生成的特征向量。替换跟踪根据特征向量的分量是 1 还是 0 来定义:

1 如果 xi,t = 1 zi, 否则 t−1。(12.12)

如今，我们发现，将痕迹替换为荷兰痕迹的粗略近似值，这在很大程度上取代了它们。荷兰痕迹通常比替代痕迹表现得更好，并且有更清晰的理论基础。积累的痕迹仍然对非线性函数近似，荷兰的痕迹是不可用的。

12.6 荷兰语在蒙特卡洛学习

尽管从历史上看，资格跟踪与 TD 学习有着密切的联系，但实际上它们与此无关。事实上，资格的痕迹甚至在蒙特卡罗学习中也会出现，正如我们在本节中所展示的。我们展示了线性 MC 算法 (第 9 章)，作为一种前向视图，可以用荷兰式跟踪来推导出一种等价但计算上更便宜的后向视图算法。这是我们在这本书中明确展示的前后观点的唯一等价性。它给的一些风味等价的证明真正的在线 TD( ) 和在线 -return 算法, 但要简单得多。梯度蒙特卡罗预测算法的线性版本 (第 202 页) 进行了如下的更新，每一集的时间步骤都有一个更新:

wt + 1 . = wt +

G−w ?t xt

xt,0 t < t。(12.13)

为了简化这个示例，我们在这里假设返回的 G 是在本集末尾收到的单个奖励 (这就是为什么 G 没有被时间下标)，并且没有折扣。在这种情况下，更新也被称为最小均方 (LMS) 规则。作为一种蒙特卡罗算法，所有的更新都取决于最终的奖励/回报，所以在这一集结束之前都不能进行任何更新。MC 算法是一种离线算法，我们不寻求改进它的这一方面。相反，我

们只是寻求一种具有计算优势的算法的实现。我们仍将只在本集结束时更新权重向量，但我们将在本集的每一步中进行一些计算，而在本集结束时进行更少的计算。这将使计算的分布更加平均——o (d) 每步，并消除了在每个步骤中存储特征向量的需要，以便在每一集结束时使用。相反，我们将引入一个额外的向量内存，称为资格跟踪，保留到目前为止看到的所有特征向量的摘要。这将足以有效地重新创建与 MC 序列完全相同的整体更新

更新 (12.13), 年底这一事件:wT = −1 +

G−w ?T−1 xt−1

xT−1

= −1 + xT−1

−x ?T wt−−1 1

+ GxT−1

=

我 − xT−1 x ?T−1

wT−1 + GxT−1 = 英尺 −1 wt−1 + GxT−1 在英国《金融时报》。= 我 − xtx 吗?t 是一个遗忘，或衰落，矩阵。现在, 递归, = 英尺 −1(英尺 −2 wt−2 + GxT−2)+ GxT−1 = 英尺 2 wt−−−1 英尺 2 + G(英尺 −1 xT−2 + xT−1) = 2 英尺 −−1 英尺 (英国《金融时报》3 wt−−3 + GxT−3)+ G(英尺 −1 xT−2 + xT−1) = 英尺 −−1 英尺 2 英尺 −3 −3 + G(英尺 −−1 英尺 2 xT−−3 + 英尺 1 xT−2 + xT−1)

=−1 + GzT−1 (12.14) −1 和 zT 型 −1 的值在时间 T−1 的两个 auxilary 记忆向量可以没有知识的不断更新的时间复杂度为 O(d) 的 G 和每一步。zt 向量实际上是一个荷兰式的资格跟踪。它被初始化为 z0 = x0，然后根据。

zt 型。= t ?k = 0 FtFt−1 ••• 颗 + 1 xk,1 t < t

= t−1 ?k = 0 FtFt−1 ••• 颗 + 1 xk + xt

英国《金融时报》= t−1 ?k = 0 英尺 −−1 英尺 2 ••• Fk + 1 xk + xt 队

= Ftzt−1 + xt

=

我 − xtx 吗?t

zt 型 −1 + xt

= zt 型 −1− xtx

t zt 型 −1 + xt = zt 型 −1−

z ?t−1 xt

xt + xt

= zt 型 −1 +

1− z 吗?t−1 xt

xt,

这是荷兰人跟踪的情况下 = 1(cf Eq。12.11)。将 at 辅助向量初始化为 a0 = w0，然后根据

在. = FtFt−1 ••• F0w0 = Ftat−1 =−1− xtx 吗?t−1,1 t < t。

辅助向量 at 和 zt 在每个时间步 t < t 时更新，然后在观察 G 时，在 (12.14) 中使用它们来计算 wT。通过这种方式，我们实现了与 MC/LMS 算法完全相同的最终结果，该算法的计算属性较差 (12.13)，但现在我们使用的是增量算法，其每一步的时间和内存复杂度为 O(d)。这是令人惊讶和有趣的，因为资格跟踪 (特别是荷兰的跟踪) 的概念出现在一个没有时间差异 (TD) 学习的环境中 (与 van Seijen 和 Sutton, 2014)。似乎资格的痕迹并不是 TD 学习所特有的; 它们比这更重要。当一个人试图以一种有效的方式学习长期的预测时，就会产生对资格的需求。

12.7 撒尔沙 ( )

本章中已经提出的思想很少需要改变，以便将可信赖性-跟踪扩展到行动-价值方法。学习近似动作值, 问ˆ(年代,w), 而不是近似状态值,v̂(s,w), 我们需要使用的行为价值形式 n-step 回报, 从第十章:

Gt:t + n。= Rt + 1 + •••+ n−1 Rt + n + nq̂(圣 + n + n,wt + n−1),t + n < t,

Gt:t + n。如果 t + n t = Gt。使用这个, 我们可以形成的行为价值形式截断 -return, 否则相同的状态值形式 (12.9)。的行为价值形式离线 -return 算法 (12.4) 仅仅使用 q̂, 而不是 v̂:

wt + 1 . = wt +

G t− 问ˆ(圣,wt)

wt 问ˆ(圣),t = 0,。、T−1(12.15)

在 G t 。= G t:∞。这个 forward 视图的复合备份关系图显示在下面图 12.9。注意到相似的图 TD( ) 算法 (图 12.1)。第一个更新将向前跨一个完整的步骤，到下一个状态-动作对，第一个更新向前两个步骤，到第二个状态-动作对，等等。最后的更新基于完整的返回。每个 n-step 的权重更新 -return 同样在 TD( ) 和 -return 算法 (12.3)。temporal-difference 方法行动值, 称为撒尔沙 ( ), 接近这个视图。它具有相同的更新规则正如前面给出的 TD( ):

wt + 1 . = wt + tzt,

当然，除了使用 TD 错误的动作-值形式:

t。= Rt + 1 + q̂(圣 + 1 + 1,wt)− 问ˆ(圣, 在 wt), (12.16)

以及合格跟踪的行动价值形式:

z −1 。= 0, zt 型。q = zt−1 +ˆ(圣, 在 wt),0 t t。

图 12.9: 撒尔沙图 ( ) 的备份。与图 12.1。

完整的伪代码撒尔沙 ( ) 和线性函数近似, 二进制特征, 并给出积累或取代痕迹的盒子在下一页。这个伪代码突出显示了二进制特性 (特性要么是活动的 (=1)，要么是不活动的 (=0)) 的一些可能的优化。示例 12.1: 在 Gridworld 中，使用合格跟踪可以大大提高控制算法对单步方法甚至 n 步方法的效率。原因如下面的 gridworld 示例所示。

第一个面板显示了代理在单个事件中采取的路径。初始估计值为零，所有的奖励都为零，除了目标位置为 g 的正奖励之外。其他面板上的箭头显示，对于各种算法，在达到目标后，动作值会增加，并增加多少。一步方法只增加最后一个动作的值，而 n 步方法将同样增加最后 n 个动作的值，而一个合格跟踪方法将更新所有动作的值，直到事件的开始，以不同的程度，随着时间的流逝而递减。衰落的策略往往是最好的。

练习 12.6 修改撒尔沙的伪代码 ( ) 使用荷兰痕迹 (12.11) 没有其他特色的一个真正的在线算法。假设线性函数近似和二进制特征。? 例 12.2: 撒尔沙 ( ) 山车图 12.10(左) 在下一页显示结果与撒尔沙 ( ) 在山上汽车任务在例 10.1 中引入的。函数逼近、动作选择和环境细节与第 10 章完全一样，因此，用数字比较这些结果与第 10 章的 n 步 Sarsa(图右侧) 的结果是恰当的。早些时候结果不同长度 n 更新而在这里撒尔沙 ( ) 我们不同跟踪参数 , 扮演类似的角色。撒尔沙的 fading-trace 引导战略 ( ) 似乎在这个问题上更高效的学习。还有我们的理想的行为价值版本 TD 方法, 在线 -return 算法 (12.4 节) 和有效实现真正的在线 TD( )(12.5 节)。在第 12.4 节中，除了使用在当前部分开始时给出的 n 步返回的动作值形式外，其余的部分都没有改变。第 12.5 和 12.6 节的分析也为行动值进行了分析，唯一的变化是使用。

撒尔沙 ( ) 取代痕迹 n-step 撒尔沙

× 瓷砖的数量 (8) × 瓷砖的数量 (8)

图 12.10: 早期性能在山上撒尔沙车的任务 ( ) 替换痕迹和 n-step 撒尔沙 (复制从图 10.4) 作为一个函数的步长, 。

状态-动作特征向量为 xt = x(St, At)，而不是状态特征向量 xt = x(St)。由此产生的高效算法的伪代码, 称为真正的在线撒尔沙 ( ) 在盒子在下一页。下图比较各种版本的撒尔沙的性能 ( ) 在山上汽车的例子。

× 瓷砖的数量 (8)

图 12.11: 总结比较撒尔沙 ( ) 算法在山上车任务。真正的在线撒尔沙 ( ) 表现的更好比普通撒尔沙 ( ) 积累和取代的痕迹。还包括是一个版本的撒尔沙 ( ) 取代痕迹, 在每个时间步, 不选择状态和行为的痕迹被设置为 0。

最后, 还有一个截断版本的撒尔沙 ( ), 称为向前撒尔沙 ( )(van Seijen,2016), 这似乎是一个特别有效的模范自由控制方法结合多层人工神经网络使用。

12.8 变量 和

我们现在开始着手开发基本的 TD 学习算法。要以最一般的形式呈现最终的算法，可以将自举和折现的程度从常数参数推广到可能依赖于状态和动作的函数。也就是说, 每个时间步将有不同的 和 , 表示 t t。现在我们改变符号, :S×→[0,1] 现在是一个函数从州和行动单位间隔, 这样 t 。= (St) 同样, :S→[0,1] 是一个函数从国家到单位间隔, 这样 t。= (St)。引入函数 , 终止功能, 尤其重要, 因为它改变了回报，这是一个基本的随机变量它的期望是我们想要的

估计。现在，更一般地定义回报为

Gt。= Rt + 1 + t + 1 gt + 1 = Rt + 1 + t Rt + 2 + 1 + t + 1 t Rt + 3 + 2 + t + 1 t + 2 t Rt + 4 + 3 + •••

= ∞ 吗?k = t 吗?k ? 我 = t + 1 i Rk + 1, (12.17)

为了保证和是有限的，我们需要它!∞ k = t k = 0 的概率这个定义的一个方便的方面是，它允许情景设置和它的算法以单一的经验流的形式呈现，而不需要特殊的终端状态、开始分布或终止时间。一个昔日的终端状态, 成为一个 (s)= 0, 开始转换分布。那样 (通过选择 (•) 作为一个常数在其他州) 我们可以恢复古典情景设置是一个特例。状态依赖终止包括其他预测情况，如伪终止，在这种情况下，我们寻求在不改变马尔可夫过程流的情况下预测一个量。折现收益可以被认为是这样一个量，在这种情况下，依赖于国家的终止结合了偶发性和折扣持续性的情况。(未讨论的持续病例仍然需要一些特殊治疗。) 对变量引导的泛化不是问题中的变化，比如折现，而是解决方案策略的变化。归纳影响 -returns 状态和行为。新的基于状态 -return 可以递归地写作为

G s t . = Rt + 1 + t + 1 ?(1− t + 1)v̂(wt) 圣 + 1 + t + 1 g s t + 1,(12.18) 现在我们已经添加了“s”上标 提醒我们, 这是一个返回接连的状态值, 区分它的返回从行动的价值观, 引导我们与“a”下面的上标。这个方程表示, -return 是第一个奖励, 尚未完全和不受引导, 加上可能连任, 我们没有折扣在下次状态 (也就是说, 根据 t + 1; 回想一下，如果下一个状态是终端，这是零。在某种程度上，我们没有在下一个状态终止，我们有第二个项它本身被划分成两种情况取决于状态的自举程度。在一定程度上引导, 这一项的估计价值, 然而, 在某种程度上, 我们不引导, 术语是下次的 -return 一步。基于动作的 -return 是撒尔沙形式

G a t . = Rt + 1 + t + 1
(1− t + 1) 问^(圣 + 1 + 1,wt)+ t + 1 g a t + 1
(12.19)

或者预期的 Sarsa 形式，

G a t . = Rt + 1 + t + 1
(1− t + 1)∇t(圣 + 1)+ t + 1 g a t + 1
, (12.20)

(7.8) 是广义函数近似为 ∇t(年代)。=
一个 (|) 问^(年代,wt)。(12.21)

12.9。带控制变量的脱机跟踪 309 年

练习 12.7 将上面的三个递归方程归纳为它们的截断版本，定义 G s t h 和 G a t:h。?

12.9 带有控制变量的非政策跟踪

最后一步是合并重要性抽样。与 n-step 方法的情况下, 完全 non-truncated -returns 一个没有一个实际的选择的重要性抽样是目标回报之外完成的。相反，我们直接使用控制变量 (7.4) 对每个决策重要性抽样的引导一般化。在状态的情况下, 我们最终 -return 概括的定义 (12.18),(7.13), 模型后

G s t .

= t

Rt + 1 + t + 1 ?(1− t + 1)v̂(wt) 圣 + 1 + t + 1 g s t + 1

+(1− t)v̂(圣、wt)(12.22)

t = (圣)| b(| St) 是通常的单步重要性抽样比例。就像我们在书中看到的其他回报一样，这个回报的截断版本可以简单地近似于基于状态的 TD 错误的总和，

s t . = Rt + 1 + t + 1 v̂(wt) 圣 + 1−v̂(St,wt), (12.23)

作为

G s t v̂(圣、wt)+ t ∞ 吗?k = t s k k ? 我 = t + 1 i i i (12.24)

当近似函数不变时，近似趋于精确。练习 12.8 证明 (12.24) 如果值函数不改变，就变得准确。为了节省写作，考虑 t = 0 的情况，并使用符号 Vk 。w = v̂(Sk)。? 12.9 运动的截断版本一般 off-policy 返回 G s t 来标示:h。根据 (12.24) 猜出正确的方程。? 上述形式的 -return(12.24) 就能够很方便的使用在一个前瞻更新、wt + 1 = wt +

wt G s t−v̂(圣)

wt v̂(圣)

wt + t

∞ 吗?k = t s k k ? 我 = t + 1 i i i wt v̂(圣),

对于有经验的人来说，这个产品看起来就像一个基于 eligibilityt 的更新——这个产品就像一个合格的跟踪，它被乘以 TD 错误。但这只是向前发展的时间步骤。我们正在寻找的关系是，对时间求和的 forward-view 更新，大约等于对时间求和的 back -view 更新 (这个关系只是近似的，因为我们再次忽略了值的变化)

功能)。前瞻性更新随时间的总和是

∞ 吗?t = 1 (wt + 1−wt) ∞ 吗?t = 1 ∞ 吗?k = t wt t s k v̂(圣) k ? 我 = t + 1 i i i

= ∞ 吗?k = 1 k ?t = 1 t v̂(圣 wt) s k k ? 我 = t + 1 i i i

(使用求和规则:?y t=x ? y k = t = k ? y = x ? k t = x)

= ∞ 吗?k = 1 s k k ?t = 1 wt t v̂(圣) k ? 我 = t + 1 i i i,

如果可以将剩下的第二个和中的整个表达式作为资格跟踪进行增量编写和更新，那么将以向后视图 TD 更新的总和的形式进行更新。我们表明, 如果 k 时刻跟踪这个表达式, 然后我们可以从它的值更新它在时间的 k−1:

zk = k ?t = 1 wt t v̂(圣) k ? 我 = t + 1 i i i

= k−1 ?t = 1 wt t v̂(圣) k ? 我 = t + 1 i i i + k v̂(Sk、工作)

其中，将索引从 k 更改为 t，是状态值的一般累加跟踪更新:

zt 型.

= t

t tzt−1 + v̂(圣 wt)

, (12.25)

这个资格的跟踪, 以及通常的 semi-gradient parameter-update 规则 TD( )(12.7), 形成一个通用 TD( ) 算法, 可以应用于对政策或 off-policy 数据。在政策的情况下, 该算法正是 TD( ) 因为 t 总是 1 和 (12.25) 成为平时积累跟踪 (12.5)(扩展到变量 和)。在非策略情况下，该算法通常工作得很好，但作为一种半梯度方法，不能保证稳定。在接下来的几节中，我们将考虑对它进行扩展，以确保稳定性。一个非常类似的一系列步骤可以按照推导出 off-policy 资

格行为价值方法和相应的痕迹一般撒尔沙 ( ) 算法。一个可以开始递归形式一般基于动作的 -return(12.19) 或 (12.20), 但后者 (预期的撒尔沙形式) 是简单的工作。我们延长 (12.20) 到

在模型 (7.14) 之后的离线情况下生产

G a t . = Rt + 1 + t + 1

(1− t + 1)∇t(圣 + 1)+ t + 1 ? t t + 1 + V + 1 g āt(圣 + 1)− t̂+ 1 q(圣 + 1 + 1,wt)

= Rt + 1 + t + 1

∇t(圣 + 1)+ t + 1 t + 1 ? G a t + 1− 问ˆ(圣 + 1 + 1,wt)

(12.26)

圣 + ∇t(1) 是由 (12.21) 给出。又可以编写 -return 大约 TD 错误的总和,

G a t 问ˆ(圣,wt)+ ∞ 吗?k = t a k k ? 我 = t + 1 i i i, (12.27)

使用基于活动的 TD 错误的期望形式:

a t = Rt + 1 + t + 1 v̄t(圣 + 1)− 问ˆ(圣, 在 wt)。(12.28)

如前所述，如果近似值函数不变，则逼近将变得精确。练习 12.10 证明 (12.27) 如果值函数不变，就会精确。为了节省写作, 考虑 t = 0 的情况下, 使用符号 Qk = 问ˆ(Sk,Ak,w)。提示: 先写出 a 0 和 G a 0, 然后 G a 0−Q0 处。? 12.11 运动的截断版本一般 off-policy 返回 G a t 来标示:h。根据 (12.27) 猜出它的正确方程。? 使用完全类似于状态的步骤，可以基于 (12.27) 编写一个前视图更新，使用求和规则来转换更新的和，并最终得出以下形式的行为值的资格跟踪:

zt 型. q = t t tzt−1 +ˆ(圣, 在 wt)。(12.29)

这个资格的跟踪, 再加上预想 TD 错误 (12.28) 和通常的 semi-gradient parameter-update 规则 (12.7), 形成一个优雅的、高效的预期撒尔沙 ( ) 算法, 可以应用于对政策或 off-policy 数据。它可能是目前这种类型的最佳算法 (当然，在以某种方式与下面几节中介绍的方法结合之前，它不能保证是稳定的)。在政策情况下常数 和 , 和通常的政府行动 TD 错误 (12.16), 该算法将是相同的撒尔沙 ( ) 算法在 12.7 节。练习 12.12 详细列出了上述从 (12.27) 推导 (12.29) 的步骤。开始更新 (12.15), 替代 G a t 从 G (12.26), 然后相似步骤如下 (12.25)。? 在 = 1, 这些算法成为相应的蒙特卡罗算法密切相关。人们可能会认为，情景性问题和离线更新将具有确切的等价性，但事实上，这种关系更微妙，也更弱。在这些最有利的条件下，仍然没有一集接一集的更新，只有他们的期望。这并不像这些方法那样令人惊讶

在轨迹展开时进行不可撤销的更新，而真正的蒙特卡罗方法在目标策略下，如果轨迹内的任何动作都是零概率的，则不会对轨迹进行更新。特别是, 所有这些方法, 即使在 = 1, 还引导, 他们的目标依赖于当前值估计, 只是依赖消掉的期望值。这在实践中是好还是坏是另一个问题。最近，提出了实现精确等价的方法 (Sutton, Mahmood, Precup and van Hasselt, 2014)。这些方法需要一个额外的“临时权重”向量来跟踪已更新的内容，但可能需要根据以后采取的行动收回 (或强调) 这些内容。国家和政府行动版本的这些方法被称为输配电 ( ) 和 PQ( ) 分别在“P”代表的是临时的。所有这些新的政策外方法的实际后果尚未确定。毫无疑问，在使用重要性抽样的所有非策略方法中都会出现高方差的问题 (第 11.9 节)。如果 < 1, 那么所有这些 off-policy 算法涉及引导和致命的三合会应用 (11.3 节), 这意味着他们只能对于表格的情况, 保证稳定的聚合状态, 和其他有限形式的函数近似。对于线性和更一般的函数逼近形式，参数向量可能会发散到无穷大，就像第 11 章中的例子一样。正如我们在那里讨论过的，离线学习的挑战有两个部分。策略之外的合格跟踪有效地处理了挑战的第一部分，修正了目标的期望值，但完全没有处理挑战的第二部分，这与更新的分发有关。第 12.11 节总结了利用资格跟踪来应对脱机学习挑战的第二部分的算法策略。练习 12.13 off-policy 的 duch -trace 和 replacing-trace 版本是什么对状态值和动作值方法的资格跟踪? ?

12.10 沃特金斯的 Q( )Tree-Backup( )

多年来已经提出了几种方法来将 Q-learning 扩展到合格的跟踪。原来是沃特金斯的 Q( ), 衰减其资格痕迹以通常的方式只要一个贪婪的行动, 然后削减为零后第一个贪婪的操作痕迹。备份图沃特金斯的 Q( ) 如图 12.12 所示。在第六章中, 我们统一 q 学习的和预期的撒尔沙 off-policy 版本的后者, 其中包括 q 学习作为一个特例, 并推广到任意目标政策, 本章在前面的小节中我们完成了治疗预期撒尔沙通过泛化 off-policy 资格痕迹。然而，在第七章中，我们区分了 n 步期望 Sarsa 和 n 步树备份，后者保留了不使用重要抽样的性质。仍然是那么的资格跟踪版本树备份, 我们将称之为 Tree-Backup( ), 或简称为结核病 ( )。这可以说是 Q-learning 的真正继承者，因为它保留了其吸引人的不重视抽样，即使它可以应用于非政策数据。

n−1 图 12.12: 备份图沃特金斯的 Q( )。组件更新系列结束要么在这一集的结尾，要么在第一个非贪婪的动作中，哪个先出现。

结核病 ( ) 的概念很简单。作为其备份图如图 12.13 所示,tree-backup 更新每个长度 (从 7.5 节) 的加权以通常的方式依赖于引导参数 。详细的方程, 与正确的指标一般引导和折扣参数, 最好先递归形式 -return 使用行动值 (12.20), 然后扩大后的目标引导的情况下的模型 (7.16):

G a t . = Rt + 1 + t + 1

(1− t + 1)∇t(圣 + 1)+ t + 1

在 + 1 ? = (| 圣 + 1) 问^(圣 + 1,wt)+ (+ 1 | 圣 + 1)G a t + 1

= Rt + 1 + t + 1

∇t(圣 + 1)+ t + 1 (+ 1 | 圣 + 1)

G a t + 1− 问^(圣 + 1 + 1,wt)

按照通常的模式，它也可以近似地 (忽略近似值函数的变化) 写成 TD 错误的总和，

G a t 问^(圣,wt)+ ∞ 吗?k = t a k k ? 我 = t + 1 i i (Ai | Si),

使用基于活动的 TD 错误的期望形式 (12.28)。与上一节相同的步骤，我们到达了一个特殊的资格跟踪更新，涉及所选操作的目标策略概率，

zt 型。在 | = t t (St)zt 型 −1 + 问^(圣, 在 wt)。

314 年第十二章: 合格的痕迹

图 12.13: 树的 版本备份的备份图算法。

加之通常 parameter-update 规则 (12.7), 定义了结核病 ( ) 算法。像所有 semi-gradient 算法, 结核病 ( ) 不保证稳定在使用 off-policy 数据和一个强大的函数近似者。获得这些保障, 结核病 ( ) 必须加上在下一小节中介绍的方法之一。∗ 锻炼 12.14 双预期如何撒尔沙延长资格的痕迹??

12.11 带有跟踪的稳定脱机方法

已经提出了几种使用合格跟踪的方法来实现脱机训练下的稳定性保证，这里我们用这本书的标准表示法提出了四个最重要的方法，包括一般的自举和贴现函数。所有这些都基于第 11.7 和 11.8 节中提出的渐变- td 或强调- td 思想。所有的算法都假设线性函数近似，虽然也可以在文献中找到非线性函数近似的扩展。GTD( ) 是类似于 TDC eligibility-trace 算法, 更好的的两个州值 Gradient-TD 预测算法在 11.7 节讨论。它的目标是学习一个参数 wt 这样 v̂(s,w)。= w ?t x(s) v (s), 甚至从数据, 是由于后另一个政策 b。其更新

wt + 1 。= wt + s t zt 型 − t + 1(1− t + 1)

z ?t vt

xt + 1,

s t、zt 型和 t 以通常的方式定义状态值 (12.23)(12.25)(11.1), 和

vt + 1 。= vt + s t zt 型 −

v ?t xt

xt, (12.30)

在 11.7 节中,v Rd 一样是一个向量的维度 w, 初始化为 v0 = 0, 和 > 0 是第二步长参数。《GQ》( ) 行动是 Gradient-TD 算法值与资格痕迹。它的目标是学习一个参数 wt 这样问ˆ(年代,wt)。= w ?t x(,) q 从 off-policy 数据 (年代)。如果目标政策 -greedy 或偏向贪婪的政策问ˆ, 那么《GQ》( ) 可以用作控制算法。它的更新

wt + 1 。= wt + a t zt 型 − t + 1(1− t + 1)

z ?t vt

x̄t + 1,

x̄t 是圣目标政策下的平均特征向量,

x̄t 。=

一个 (| St)x(圣,),

a t 是期望形式的 TD 错误, 可以写

a t . = Rt + 1 + t + 1 w ?t x̄t + 1−w ?t xt, zt 型定义以通常的方式行动值 (12.29), 和其他在 GTD( ), 包括更新 vt(12.30)。HTD( ) 是一个混合的州值算法的结合方面 GTD( ) 和 TD( )。最吸引人的特征是, 它是一种严格的泛化 TD( )off-policy 学习, 这意味着如果行为政策是一样的目标政策, 然后 HTD( ) 一样成为 TD( ), 这对 GTD( ) 是不正确的。这是吸引人的, 因为 TD( ) 通常是速度比 GTD( ) 两种算法收敛时, 和 TD( ) 只需要设置一个步长。HTD( ) 被定义为

wt + 1 . = wt + s t zt 型 + ?(zt 型 −zb t)?vt(xt− t + 1 xt + 1), vt + 1 。= vt + s t zt 型 −

zb t

vt

(xt− t + 1 xt + 1), 半。= 0, zt 型.

= t

t tzt−1 + xt

, 与 z −1 。= 0, zb t . = t tzb t−1 + xt,zb−1 。= 0,

其中 > 0 又是第二步长参数。除了第二组重量、vt,HTD( ) 也有第二组资格痕迹,zb t。这些都是传统的积累资格的行为痕迹政策, 成为等于 zt 型如果所有 t 1, 导致最后一个学期 wt 更新为零和整体更新来减少 TD( )。强调 TD( ) 的扩展一步 Emphatic-TD 算法 (章节 9.11 和 11.8) 资格的痕迹。结果算法保留了很强的非策略收敛保证，同时允许任何程度的自举，尽管代价是

方差大，收敛速度慢。强调 TD( ) 被定义为

wt + 1 . = wt + tzt t。= Rt + 1 + t + 1 w ?t xt + 1−w ?t xt zt 型.

= t

t tzt−1 + Mtxt

, 与 z −1 。= 0, 太。= t +(1− t) 英国《金融时报》英国《金融时报》. = t−1 tft−1 +,F0 。= 我 (S0),

那里太 0 的一般形式是强调, 英国《金融时报》 0 称为 followon 跟踪, 0 是利息, 如 11.8 节所述。注意, 太 t 一样, 并不是真正的一个额外的内存变量。通过将其定义代入可列性-迹方程，可以将其从算法中删除。真正的在线版本的伪代码和软件 Emphatic-TD( ) 是 web 上可用 (萨顿,2015 b)。在政策的情况下 (t = 1,t)Emphatic-TD( ) 是类似传统 TD( ), 但仍显著不同。事实上, 而 Emphatic-TD( ) 是保证为所有情境依靠 函数收敛,TD( ) 不是。TD( ) 保证收敛只有常数 。参见 Yu 的反例 (Ghiassian, Rafiee, Sutton, 2016)。

12.12 实现问题

乍一看，使用资格跟踪的列表方法可能比单步方法复杂得多。一个幼稚的实现将要求每个状态 (或状态操作对) 在每个时间步骤上更新其值估计和它的资格跟踪。对于单指令、多数据、并行计算机或似是而非的人工神经网络 (ANN) 实现来说，这不是一个问题，但对于常规串行计算机的实现来说，这是一个问题。幸运的是, 对于典型值 和 资格的痕迹几乎所有州几乎总是几乎为零; 只有最近访问过的那些状态的跟踪值将显著大于 0，只有这些很少的状态需要更新以接近这些算法。因此，在实践中，常规计算机上的实现可能只跟踪和更新明显大于零的少数跟踪。使用这个技巧，在表格方法中使用跟踪的计算开销通常是单步方法的几倍。确切的多个当然取决于 和 和其他费用的计算。请注意，表格情况在某种意义上是资格跟踪计算复杂性的最坏情况。当使用函数逼近时，不使用跟踪的计算优势通常会降低。例如，如果使用了 ANNs 和反向传播，那么资格跟踪通常只会导致每一步所需的内存和计算加倍。截断 -return 方法 (12.3 节) 可以在传统计算机计算效率虽然他们总是需要一些额外的内存。

12.13 结论

与 TD 错误结合的资格跟踪提供了一种有效的、递增的在蒙特卡罗和 TD 方法之间进行转换和选择的方法。第 7 章的 n 步方法也支持这一点，但是资格跟踪方法更通用，学习起来通常更快，并且提供不同的计算复杂度权衡。本章介绍了关于政策内外学习和变量自举和折现的资格跟踪的优雅、新兴的理论理解。这个优雅的理论的一个方面是真正的在线方法，它精确地再现了昂贵的理想方法的行为，同时保持了传统的 TD 方法的计算一致性。另一个方面是派生的可能性，它可以自动地从直观的前视图方法转换为更有效的增量后视图算法。我们用一种经典的、昂贵的蒙特卡罗算法推导出了这一总体思路，并以一种廉价的增量非 TD 实现方式结束，使用的是真正的在线 TD 方法中使用的同一种新的合格跟踪。正如我们在第 5 章中提到的，蒙特卡罗方法在非马尔可夫任务中可能有优势，因为它们没有引导。由于资格跟踪使 TD 方法更像蒙特卡罗方法，因此它们在这些情况下也具有优势。如果由于 TD 方法的其他优点而希望使用 TD 方法，但是任务至少是部分非马尔可夫的，那么就需要使用资格跟踪方法。资格跟踪是针对长期延迟的奖励和非马尔可夫任务的第一道防线。通过调整 , 我们可以将资格沿着连续跟踪方法在任何地方从蒙特卡罗一步 TD 方法。我们把它们放在哪里? 对于这个问题，我们还没有一个好的理论答案，但一个明确的经验答案似乎正在出现。对于每集有许多步骤的任务，或者在折现的半衰期内有许多步骤的任务，使用资格跟踪比不使用要好得多 (例如，参见图 12.14)。另一方面，如果痕迹长到产生一个纯粹的蒙特卡罗方法，或者差不多，性能就会急剧下降。中间混合物似乎是最好的选择。资格的痕迹应该用来带我们走向蒙特卡罗方法，但不是一直那样。在将来它可能会更多的精细变化之间的权衡 TD 和蒙特卡罗方法通过使用变量 , 但目前尚不清楚这是如何做到的可靠和有效。使用资格跟踪的方法需要比一步法更多的计算量，但反过来它们提供了显著更快的学习速度，特别是当奖励被许多步骤延迟时。因此，在数据稀缺且无法重复处理的情况下，使用资格跟踪通常是有意义的，这在在线应用程序中经常出现。另一方面，在离线应用程序中，数据可以很便宜地生成，可能来自廉价的模拟，然后通常不支付使用资格跟踪。在这些情况下，目标不是从有限的数据中获得更多，而是尽可能快地处理尽可能多的数据。在这些情况下，由于跟踪而导致的每个数据的加速率通常不值得它们的计算成本，并且支持一步法。

0 0.2 0.4 0.6 0.8 第十二章: 合格的痕迹

图 12.14: 在强化学习性能的影响在四个不同的测试问题。在所有情况下, 性能通常是最好的 (图中较低的数量) 的中间值 。左边的两个面板是简单的连续状态控制的应用程序任务使用撒尔沙 ( ) 算法和瓷砖编码、替换或积累痕迹 (萨顿,1996)。政策评估的右上角面板随机漫步的任务使用 TD( )(辛格和萨顿,1996)。右下角的面板是平衡杆任务的未发布数据 (例子 3.4) 来自早期的研究 (Sutton, 1984)。

12.14 书目的和历史的言论

通过 Klopf(1972) 的丰富思想，获得了资格证书。我们使用资格追溯的依据是 Klopf 的工作 (Sutton, 1978a, 1978b, 1978c;Barto 和 Sutton, 1981a, 1981b; 萨顿和 Barto,1981; 巴托，萨顿，安德森，1983 年; 萨顿,1984)。我们可能是第一个使用“资格追踪”一词的 (Sutton and Barto, 1981a)。刺激物在神经系统中产生后效应对学习很重要，这一观点由来已久 (见第 14 章)。第 13 章讨论了演员-批评家的方法 (Barto, Sutton, and Anderson, 1983; 萨顿,1984)。

12.1 复合更新在第一版中称为“复合备份”书。 -return 及其错误降低属性引入了沃特金斯 (1989) 和由 Jaakkola 进一步发展, 约旦和辛格 (1994)。在这个和后面的部分中，random walk 的结果是新的，比如“forward view”和“向后视图”。“ -return 算法的概念引入本文的第一版。这里提出的更精细的处理方法是与危害范思珍 (例如，van Seijen 和 Sutton, 2014) 一起开发的。

12.2 TD( ) 积累了痕迹, 萨顿 (1988、1984)。共同大研 (1992) 证明了平均值的收敛性，许多研究人员都有概率 1，包括 Peng(1993)、Dayan 和 Sejnowski(1994)、Tsitsiklis(1994)、Gurvits、Lin 和 Hanson(1994)。绑定的误差渐近 -dependent 解决线性 TD( ) 是由于 Tsitsiklis 和范 • 罗伊 (1997)。

12.3 截断的 TD 方法由 Cichosz(1995) 和 van Seijen(2016) 开发。12.4 重做更新的想法最初是由 van Seijen 提出的以“最佳比赛学习”为名 (van Seijen, 2011;van Seijen, Whiteson, van Hasselt 和 weling2011)。

12.5 真实在线 TD( ) 主要是由于伤害 van Seijen(van Seijen 和萨顿, 2014;van Seijen et al.，2016) 尽管它的一些关键思想是由 Hado van Hasselt(个人交流) 独立发现的。“荷兰痕迹”这个名字是为了表彰两位科学家的贡献。替换痕迹要归功于辛格和萨顿 (1996)。

12.6 本节材料来自 van Hasselt 和 Sutton(2015)。

12.7 撒尔沙 ( ) 与积累痕迹最初探索的控制方法酒店和 Niranjan(1994; 酒店,1995)。真正的在线撒尔沙 ( ) 介绍了 van Seijen 和萨顿 (2014)。第 307 页的算法是

改编自 van Seijen et al.(2016)。山车的结果是为本文制作的，除了图 12.11 改编自 van Seijen 和 Sutton(2014)。

12.8 也许第一个发表讨论变量 是沃特金斯 (1989) 指出, 更新的切断序列 (图 12.12) 在他问 ( ) 当 nongreedy 行动选择可以实现通过暂时设置 为 0。介绍了变量 第一版的文本。变量 的根是在工作上选项 (Precup 萨顿, 辛格 1999) 及其前体 (萨顿,1995), 成为《GQ》( ) 显式纸 (Maei 和萨顿,2010), 也介绍了一些 -returns 递归形式。不同的变量概念 开发了玉 (2012)。

12.9 随后 Precup 等 (2000 年，2001 年) 引入了保单之外的资格追溯由 Bertsekas 和 Yu (2009)，Maei (2011;Maei 和 Sutton, 2010)，Yu (2012)，Sutton, Mahmood, Precup, van Hasselt(2014)。尤其是最后参考使一个强大的前锋视图 off-policy TD 方法依赖政府与通用 和 。这里的报告似乎是新的。本节以一个优雅的预期撒尔沙 ( ) 算法。虽然它是一种自然算法，但据我们所知，它还没有在文献中被描述或测试过。

12.10 沃特金斯的 Q( ) 是由于沃特金斯 (1989)。表格式的，情节式的，离线的版本 Munos、Stepleton、Harutyunyan 和 Bellemare(2016) 证明了收敛性。替代 Q( ) 算法提出的彭和威廉姆斯 (1994、1996) 和萨顿, 马哈茂德,Precup, 范特 (2014)。树备份 ( ) 是由于 Precup, 萨顿, 辛格 (2000)。

12.11 GTD( ) 是由于 Maei(2011)。《GQ》( ) 是由于 Maei 和萨顿 (2010)。HTD( ) 是由于白色和白色 (2016) 基于一步 HTD 算法引入了哈克曼 (2012)。梯度- td 方法理论的最新发展是 Yu(2017)。强调 TD( ) 引入的萨顿, 马哈茂德, 和白色 (2016), 证明了其稳定性。Yu(2015, 2016) 证明了其收敛性，算法由 Hallak et al.(2015, 2016) 进一步开发。

13. 第十三章策略梯度方法

在这一章中，我们考虑一些新的东西。到目前为止，这本书中几乎所有的方法都是行动价值方法; 他们学习行动的价值，然后根据估计的行动价值选择行动; 如果没有行动价值的估计，他们的政策甚至不会存在。在本章中，我们考虑的方法是学习参数化策略，该策略可以在不考虑值函数的情况下选择操作。值函数仍然可以用于学习策略参数，但不需要进行操作选择。我们使用符号 *Rd* 吗? 用于策略的参数向量。因此我们写 *(|* 年代, *)=* 公关 *= | 圣 = s, t =* 的概率是采取行动在时间 *t* 给定状态的环境在时间 *t s* 参数 。如果一个方法使用了价值函数, 那么价值函数的权向量 *w* 来标示 *Rd* 像往常一样, 在 *v̂(s,w)*。在这一章里, 我们考虑学习策略参数的方法基于一些标量性能的梯度测量 *J( )* 的政策参数。这些方法试图最大化性能，所以它们的更新近似于 *J* 的梯度上升:

t + 1 = t + ? J(t), (13.1) 在哪里*? J(t) Rd* 随机估计的预期接近性能的梯度测量的参数 *t*。所有遵循这个通用模式的方法都称为策略梯度方法，不管它们是否也学习一个近似值函数。学习逼近策略和值函数的方法通常被称为 *actor - critics* 方法，其中 '*actor*' 是学习策略的参考，'批评家' 是学习的值函数，通常是状态值函数。首先，我们讨论情景情形，其中性能被定义为参数化策略下的开始状态的值，然后再考虑继续情形，其中性能被定义为平均回报率，如第 *10.3* 节所示。最后，我们可以用非常相似的术语表示这两种情况的算法。

唯一的例外是 *2.8* 节的梯度班迪特算法。事实上，这一节经历了许多相同的步骤，在单状态强盗案例中，当我们在这里进行完整的 *MDPs* 时。复习这一节将是充分理解本章的充分准备。

13.1 政策近似值及其优点

在策略梯度方法, 策略可以参数化以任何方式, 只要 (| 年代,) 是可微的参数, 也就是说, 只要 (| 年代,)(列向量的偏导数的 (| 年代,) 关于 的组件) 为所有 年代存在并且是有限的, 一个 (s), 和 Rd 吗?。在实践中，为了确保探索，我们通常要求政策永远不会成为决定性的 (即: (| 年代,) (0,1), 对所有的年代, 一个,)。在本节中，我们将介绍离散操作空间中最常见的参数化，并指出它相对于操作值方法的优点。基于策略的方法还提供了处理连续操作空间的有用方法，我们将在后面的第 13.7 节中对此进行描述。如果操作空间是离散的, 而不是太大, 那么自然和常见的一种参数化是形成参数化数值偏好 h(s,a,) R 为每个政府行动。在每个状态中，具有最高偏好的操作被给定的被选择的最高概率，例如，根据指数软最大值分布:

在 e 2.71828 是自然对数的基础。注意这里的分母是要求每个状态的动作概率之和为 1。我们把这种政策参数化称为行动偏好中的软最大值。操作首选项本身可以被任意参数化。例如, 他们可能被深度计算人工神经网络 (ANN), 是所有网络的连接权值的向量 (如 16.6 节中描述的 AlphaGo 系统)。或者偏好可以是线性的，h(年代,)= ?x(,), (13.3) 使用特征向量 x(,) Rd ?

由第 9 章描述的任何方法构造。参数化的一个优点政策根据 soft-max 行动偏好是近似政策可以达到一个确定的政策, 而 -greedy 行动选择在行动总是有一个 值的概率随机选择一个行动。当然，可以根据基于动作值的软最大值分布进行选择，但仅凭这一点不能使策略接近确定性策略。相反，行动价值估计将收敛到它们相应的真实值，它们之间的差异是有限的，可以转化为除 0 和 1 以外的特定概率。如果 soft-max 分布包括温度参数, 然后随着时间的推移, 温度可以减少方法决定论, 但是在实践中很难选择减少计划, 甚至初始温度, 没有更多的先验知识的真正的行动值比我们愿意承担。行动偏好是不同的，因为它们不接近特定的值; 相反，他们被迫制定最优的随机政策。如果最优策略是确定性的，那么最优操作的首选项将被驱动到比所有次最优操作 (如果参数化允许的话) 高无穷大。

根据 action 偏好中的 soft-max 对策略进行参数化的第二个优点是，它允许选择具有任意概率的操作。在具有显著函数逼近的问题中，最佳逼近策略可能是随机的。例如，在信息不完全的纸牌游戏中，最优的玩法通常是用特定的概率做两件不同的事情，比如在扑克中虚张声势。行为-价值方法没有找到随机最优策略的自然方法，而策略近似方法可以，如示例 13.1 所示。

策略参数化可能比动作值参数化有一个最简单的优点，那就是策略可能是一个更简单的近似函数。问题在于他们的政策和行动价值职能的复杂性。对于某些人来说，动作-值函数更简单，因此更容易近似。对其他人来说，政策更简单。在后一种情况下，基于策略的方法通常会学习得更快，并产生更好的渐近策略 (如俄罗斯方块); 看到 imşek, 藻类´ 等, 和 Kothiyal,2016)。

最后，我们注意到，政策参数化的选择有时是向强化学习系统注入有关政策预期形式的先验知识的好方法。这通常是使用基于策略的学习方法的最重要的原因。

练习 13.1 利用你对 gridworld 的知识和它的动力学来确定一个精确的符号表达式，以确定选择正确动作的最佳概率。例 13.1。？

13.2 政策梯度定理

除了政策参数的实际优势 -greedy 行动选择, 还有一个重要的理论优势。连续概率变化政策参数化行动顺利学习参数的函数, 而在 -greedy 选择任意小的行动概率可能会发生戏剧性的变化估计行动值的变化, 如果变化导致不同的操作有极大价值。很大程度上由于这种更强的收敛性，政策梯度方法比行动价值方法更有效。特别是，政策依赖于参数的连续性使得政策梯度方法能够近似梯度上升 (13.1)。情景和持续的情况下定义的业绩衡量,J( ) 是不同的, 因此在某种程度上必须分别对待。尽管如此，我们将尝试统一地呈现这两种情况，并开发一个符号，以便主要的理论结果可以用一组方程来描述。在这一节中，我们讨论情节的情况，我们将性能度量定义为情节开始状态的值。我们可以简化表示法而不丢失任何有意义的通用性，假设每一集都以某种特定的 (非随机的) 状态 s0 开始。然后，在情景性的例子中我们把性能定义为

J( )。= v (s0), (13.4)

v 的真正价值函数, 的政策决定。从这里在我们的讨论, 我们将假定没有折扣 (= 1) 对于情景的情况, 虽然出于完整性的考虑我们做装箱算法包括折扣的可能性。使用函数逼近，以确保改进的方式更改策略参数似乎很有挑战性。问题是，性能既取决于操作选择，也取决于进行这些选择的状态的分布，而这两者都受到策略参数的影响。给定一个状态，策略参数对动作的影响，以及对奖励的影响，可以从参数化的知识中以相对直接的方式来计算。但政策对国家分布的影响是环境的函数，通常是未知的。当梯度依赖于政策变化对国家分布的未知影响时，我们如何能估计相对于政策参数的性能梯度？幸运的是，对于这个挑战，有一个很好的理论答案，那就是政策梯度定理，它提供了梯度的解析表达式

对于不涉及状态分布的导数的策略参数 (这是梯度上升所需的近似值 (13.1)) 的性能。情景案例的政策梯度定理证明了这一点

J( )？年代 (s)？一个 q (s) (| 年代,), (13.5)

偏导数的梯度是列向量 的组件, 和 表示相对应的政策参数向量 。这里的象征 意味着

“成正比”。在情景性的例子中，比例常数是一集的平均长度，在连续的例子中是 1，所以这个关系实际上是一个等式。这里的分布 (如第 9 和第 10 章) 在政策分布在 (见 199 页)。在上一页的方框中证明了政策梯度定理。

13.3 加强: 蒙特卡罗政策梯度

现在我们准备推出我们的第一个政策梯度学习算法。回想一下我们随机梯度上升的总体策略 (13.1)，它要求获得样本，使样本梯度的期望与性能度量的实际梯度成参数的函数成正比。样本梯度只需要梯度成正比, 因为可以吸收任何比例常数步长 , 否则是任意的。政策梯度定理给出与梯度成正比的精确表达式; 所需要的只是某种抽样方法，其期望等于或近似于这个表达式。注意右边的策略梯度定理是州加权求和的频率状态下发生的目标政策 ; 如果 是紧随其后, 那么国家会遇到这些比例。因此

J( ) ? 年代 (s)? 一个 q (s) (| 年代,)

= E

一个 q (St) (| 圣,)

. (13.6)

我们可以在这里停止并实例化我们的随机梯度上升算法 (13.1)。

t + 1 . = t +

一个问^(圣,w) (| 圣,), (13.7)

在问^q 学到一些近似。这个算法, 它被称为一个所有操作方法, 因为它更新涉及到的所有行动, 承诺和值得进一步研究的, 但是我们的活期利息是经典的增强算法在时间 t(威廉,1992) 的更新包括在 t 时刻真正采取行动。我们继续用引入 St in(13.6) 的方法来进行强化的推导，通过替换随机变量的可能值的总和。

下一个期望 , 然后抽样期望。方程 (13.6) 包括一个适当的行动求和, 但每一项不加权 (| 圣,) 需要一个期望在 。所以我们引入这样一个权重, 在不改变平等, 然后把总结项乘以 (| 圣,)。继续 (13.6)，我们有。

如往常一样，Gt 是返回。括号中的最后一个表达式正是我们所需要的，一个量可以在每个期望等于梯度的时间步上抽样。利用这个样本实例化我们的一般随机梯度上升算法 (13.1)，得到强化更新:

这个更新具有直观的吸引力。每一个增量都与一个返回的 Gt 和一个向量的乘积成比例，即采取实际行动的概率的梯度除以采取该行动的概率。矢量是参数空间中的方向，大多数增加了在未来访问状态时重复动作的概率，更新增加了这个方向的参数矢量与返回的比例，与动作概率成反比。前者是有意义的，因为它使参数朝有利于产生最高回报的操作的方向移动。后者是有意义的，因为在其他方面，经常被选择的操作具有优势 (更新将更经常地朝着它们的方向进行)，并且可能会胜出，即使它们没有产生最高的回报。注意，强化使用从时间 t 的完整返回，包括所有未来的奖励直到事件结束。从这个意义上来说，“强化”是一种蒙特卡罗算法，它只适用于情节结束后的所有更新 (就像第五章中的蒙特卡罗算法)。注意，伪代码最后一行中的更新与加固更新规则 (13.8) 有很大的不同。一个区别是, 伪代码使用紧凑表达式 ln (| 圣, t) 部分向量 (| 圣, t) (| 圣, t)(13.8)。这两个向量的表达式是等价的遵循从身份 lnx = x x .

这个向量在文献中已经有了几个名称和符号; 我们将它简单地称为资格向量。注意，这是算法中出现策略参数化的惟一位置。

第二个区别伪代码更新和加强更新方程 (13.8) 是, 前者包括 t 的因素。这是因为, 正如前面提到的, 在文本中我们把 non-discounted 案例 (= 1) 在我们给的装箱算法算法对于一般折扣情况。所有的想法都经过了适当的调整 (包括对 199 页的方框)，但是涉及到额外的复杂性，会分散对主要想法的注意力。∗13.2 概括 199 页 < 锻炼, 策略梯度公式 (13.5), 策略梯度定理

的证明 (325 页), 和步骤导致加强更新方程 (13.8), 这 (13.8) 最终的因素 t 从而符合伪代码中给出的一般算法。?

图 13.1 从示例 13.1 中显示了加固在短走廊网格世界上的性能。

-20 年

-60 -40 总回报集平均超过 100 分 G0

-80 年

-90 年 200 年 1 1000 800 600 400 集

图 13.1: 加固短走廊网格世界 (示例 13.1)。有一个很好的步骤每集的总奖励接近开始状态的最优值。

加固作为一种随机梯度法，具有良好的理论收敛性。通过构造，一个事件的预期更新与性能梯度方向相同。这保证预期性能的改善 足够小, 并收敛到局部最优标准随机近似条件下降低 。然而，作为蒙特卡罗方法的强化可能是高方差的，从而产生缓慢的学习。练习 13.3 在第 13.1 节中，我们考虑了使用 soft-max 在 action preferences(13.2) 和线性 action preferences(13.3) 的策略参数化。对于这个参数化，证明合格向量是

ln (| 年代,)= x(,)−? b (b | s)x(年代,b), (13.9)

使用定义和基本微积分。?

13.4 加强与基线

政策梯度定理 (13.5) 可以推广到包括对任意基线 b 的作用值的比较:

J( ) ? 年代 (s)? 一个

q (s)−b(s)

(| 年代,)。(13.10)

基线可以是任何函数，甚至是随机变量，只要它不随 a 变化; 这个方程仍然有效，因为减去的量是零:

一个 b(s) (| 年代,)=(s) 吗? 一个 (| 年代,)= b(s) 1 = 0。

使用带基线的策略梯度定理 (13.10) 可以使用前一节中类似的步骤派生更新规则。我们最后得出的更新规则是一个新的加固版本，包括一个通用基线:

t + 1 . = t +

Gt−b(圣) t ? (| 圣) (| 圣 t)。(13.11) 因为基线可以一致为零，所以这个更新是加固的严格推广。通常，基线使更新的期望值保持不变，但是它会对其方差产生很大的影响。例如，我们在 2.8 节中看到，类似的基线可以显著降低梯度班迪特算法的方差 (从而加速学习)。在 bandit 算法中，基线只是一个数字 (到目前为止所看到的奖励的平均值)，但是对于 MDPs，基线应该随状态而变化。在某些状态下，所有的行为都有高价值，我们需要一个高基线来区分高价值行为和低价值行为; 在其他状态下，所有的操作都有低的值和低基线是合适的。一个自然选择基线是一个估计的状态值,v̂(St,w), 其中 w Rm 是权重向量学习通过前一章中提出的方法之一。

因为加强蒙特卡罗方法对学习策略参数, , 似乎也自然地使用蒙特卡罗方法学习状态值权重,w。一个完整的强化与基线算法伪代码使用这种学习状态值函数作为基线在下面的框中给出。

该算法有两个步骤大小, 表示 和 w((13.11) 中的)。选择的步长值 (这里 w) 相对比较容易; 在线性情况下, 我们对设置的经验法则, 如 w = 0.1 / E ? ? v̂(St,w)?2 (见 9.6 节)。更清楚的是如何设置策略的步长参数, 的最佳值的偏差范围取决于奖励和政策参数化。

-60 -40 总回报集平均超过 100 分 G0

-80 年

-90 年

400 200 1 1000 800 600

集图 13.2: 添加一个基线来加强可以使它更快地学习，如下面在短走廊网格世界中所示 (示例 13.1)。这里用于普通加固的阶梯尺寸是它表现最好的 (最接近 2 的幂; 见图 13.1)。

图 13.2 比较了在短廊道网格字上有无基线的加固行为 (示例 13.1)。这里使用的近似状态值函数基线是 v̂(s,w)= w。即 w 是一个单独的组件,w。

13.5 Actor-Critic 方法

虽然增强-带基线的方法学习策略和状态-值函数，但我们不认为它是一个行为-批评方法，因为它的状态-值函数只被用作基线，而不是批评家。也就是说，它不用于引导 (从随后状态的估计值更新状态的值估计值)，而仅用于更新估计值的状态的基线。这是一个有用的区别，因为只有通过自举，我们才会引入偏差和渐近依赖于函数近似的质量。如我们所见，通过自举和依赖状态表示引入的偏差通常是有益的，因为它减少了差异并加速了学习。使用基线增强是无偏见的，将渐进地收敛到局部最小值，但是像所有蒙特卡罗方法一样，它往往学习缓慢 (产生高方差估计)，并且不方便在线实现或处理持续的问题。正如我们在本书前面所看到的，用时间差的方法可以消除这些不便，通过多步骤的方法，我们可以灵活地选择自举的程度。为了在策略梯度方法的情况下获得这些优势，我们使用了自适应批评家方法。首先考虑一步行为-批评方法，类似于第 6 章中引入的 TD 方法，如 TD(0)、Sarsa(0) 和 Q-learning。单步方法的主要吸引力在于它们是完全在线的和递增的，但是避免了资格跟踪的复杂性。它们是资格跟踪方法的一种特殊情况，不是一般的，但更容易理解。用一步回归法 (采用学习状态值函数作为基线) 代替完全回归 (13.11) 如下:

t + 1 。= t + Gt:t + 1−v̂(St,w) (| 圣, t) (| 圣 t) (13.12) = t +

Rt + 1 + v̂(w) 圣 + 1−v̂(St,w) t ? (| 圣) (| 圣 t)(13.13) = t + t (| 圣 t) (| 圣 t)。(13.14) 与此相匹配的自然状态函数学习方法是半梯度 TD(0)。完整算法的伪代码在下一页顶部的框中给出。请注意，它现在是一个完全在线的增量算法，状态、行为和奖励在发生时进行处理，然后不再重新访问。

13.6 持续问题的政策梯度

如第 10.3 节所讨论的，对于没有发作边界的持续问题，我们需要根据每次步骤的平均奖励率来定义性能:

= lim t→∞ E(t Rt | S0 A0:−1)

=

年代 (s)? 一个 (|)? r s ?, p(s ?)r,r | s,

其中 是稳态分布在 , (年代)。= limt→∞ 公关圣 = | A0:t , 这是假定存在和独立 S0(一个遍历性假设)。记住, 这是特殊的分配下, 如果您选择的行为根据 , 你留在相同的分布:

年代 (s)? 一个 (| 年代,)p(s ?| 年代,)= (s ?), 对所有年代? 年代。(13.16)

在下面的框中给出了持续情况下 (向后视图) 的 actor -批评者算法的完整伪代码。

当然, 在持续的情况下, 我们定义值,v (年代)。= E [Gt | 圣 = s] 和 q (年代)。= E [Gt | 圣 =,= 一个), 对微分返回:

Gt。= Rt + 1 Rt + 2−−r( )+( )+ Rt + 3−r( )+ •••。(13.17) 有了这些交替的定义，情景案例 (13.5) 中给出的政策梯度定理对持续案例仍然成立。在下一页的方框里有证据。前后视图方程也保持不变。

13.7 持续行动的政策参数化

基于策略的方法提供了处理大型操作空间的实用方法，甚至是具有无限多个操作的连续空间。我们学习概率分布的统计，而不是计算每一个动作的学习概率。例如，操作集可能是实数，操作选择自正态 (高斯) 分布。正态分布的概率密度函数通常是这样写的

−1 0 2 4 2−−4 0 = 0 = 0, = −2 = 2 0.2,1.0 = 2, = 2 5.0,0.5 = 2, = 、 是正态分布的平均值和标准偏差, 当然这里 是 数量 3.14159。对几种不同均值和标准差的概率密度函数向右显示。p(x) 是 x 点的概率密度，不是概率。它可以大于 1; 它是 p(x) 下的总面积，它的总和必须是 1。一般来说，我们可以取 p(x) 下的积分对于任意范围的 x 值来得到 x 落在这个范围内的概率。为了生成策略参数化，可以将策略定义为实值标量操作的正态概率密度，其均值和标准偏差由依赖于状态的参数函数逼近器给出。也就是说,

(| 年代,)。= 1 (年代,)√2 exp

− (− (年代,))2 2 (年代,)2

, (13.19)

的地方:S×Rd ?→R 和 :S×Rd 吗?→R + 两个参数化函数近似者。为了完成这个示例，我们只需要为这些逼近者提供一个表单。我们把政策的参数向量分成两部分, =[ ]?，其中一部分用于均值的近似，另一部分用于标准差的近似。均值可以近似为一个线性函数。标准差必须是正的，最好近似为线性函数的指数。因此

(年代,)。= ? x (s) 和 (年代,)。= 经验

? x (s)

, (13.20)

x (s) 和 x (s) 是国家特征向量可能由第九章中描述的方法之一。有了这些定义，本章后面描述的所有算法都可以用来学习选择实值操作。练习 13.4 表明，对于高斯政策参数化 (13.19)，资格向量有以下两部分:

ln (| 年代,)= (| 年代,) (| 年代,)= 1 (年代,)2

一个 − (年代,)x (s) 和

ln (| 年代,)= (| 年代,) (| 年代,)=

一个 − (年代,)2 (年代,)2−1 x (年代)。?

练习 13.5 bernoullil -logistic 单元是一种在某些 ANNs 中使用的随机神经元样单元 (9.6 节)。它在 t 时刻的输入是特征向量 x(St); 在, 它的输出是一个随机变量有两个值,0 和 1, 公关在 = 1 = Pt 和公关在 = 0 = 1−Pt(伯努利分布)。让 h(年代,0,) 和 h(年代,1,) 的偏好在国家年代单元的两个动作参数 既定政策。假设行动偏好之间的差异是由单元的输入向量的加权和, 也就是说, 假设 h(年代,1,)−h(年代,0,)= ? x(s), 是单元的权向量。

(一) 表明, 如果指数 soft-max 分布 (13.2) 是用来行动偏好转换为政策, 然后 Pt = (1 | 圣, t)= 1 /(1 + exp(− ?(t x(St)))(逻辑函数)

(b) 的蒙特卡罗加强更新 t t + 1 返回 Gt 收到吗?

(c) 表达资格 ln (| 年代,)Bernoulli-logistic 单元, 在一个方面,x(s), (| 年代,) 通过计算梯度。

提示: 分别为每个行动的导数计算对数首先对 Pt = (| 年代, t), 将两个结果组合成一个表达式, 取决于和 Pt, 然后用链式法则, 指出物流的导数函数 f(x) f(x)(1−f(x))。?

13.8 总结

在这一章之前，这本书关注的是行动价值方法——意思是学习行动价值，然后用它们来决定行动选择的方法。另一方面，在本章中，我们考虑了学习参数化策略的方法，该策略允许在不考虑行动价值估计的情况下采取行动。特别是，我们已经考虑了策略梯度方法——即在每个步骤上更新策略参数的方法，以估计相对于策略参数的性能梯度。学习和存储策略参数的

方法有很多优点。他们可以学习具体的行动概率。他们可以学习适当水平的探索，并渐进地接近确定的政策。它们可以自然地处理连续的操作空间。所有这些事情都是简单的基于策略的方法, 但尴尬或不可能 -greedy 一般方法和行为价值的方法。此外，在某些问题上，策略的参数表示比值函数更简单; 这些问题更适合于参数化策略方法。参数化策略方法也有一个重要的理论优势，其形式是策略梯度定理，它给出了一个精确的公式，用于描述不涉及状态分布导数的策略参数如何影响性能。这个定理为所有的政策梯度方法提供了理论基础。强化方法直接遵循政策梯度定理。增加一个状态值函数作为基线减少了加固的方差而不引入偏差。使用自引导的状态值函数引入了偏置，但通常出于同样的原因，引导 TD 方法往往优于蒙特卡罗方法 (大大减少了方差)。状态-值函数将信用赋值给批判——政策的行为选择，因此前者被称为批判者，后者被称为行动者，这些整体方法被称为行为-批判方法。总体而言，策略梯度方法提供了一组与行动价值方法截然不同的优点和缺点。如今，在某些方面，人们对它们的理解还不那么透彻，但它们却是一个令人兴奋的话题和正在进行的研究。

13.9 书目的和历史的言论

我们现在看到的与政策梯度相关的方法实际上是在强化学习中最早被研究的一些方法 (Witten, 1977; 巴托，萨顿，安德森，1983 年; 萨顿,1984; 威廉姆斯，1987 年，1992 年) 和在前人的领域 (Phansalkar 和 Thathachar, 1995 年)。在 20 世纪 90 年代，他们基本上是被这本书的其他章节所关注的行动价值方法所取代。然而，近年来，对演员-批评家方法和一般的政策-梯度方法的注意又恢复了。除了我们在这里讨论的之外，还有自然梯度法 (Amari, 1998;Kakade, 2002, Peters, Vijayakumar 和 Schaal, 2005; 彼得斯和绍尔对,2008;Park, Kim and Kang, 2005;Bhatnagar, Sutton, Ghavamzadeh 和 Lee, 2009; 参见 Grondman, Busoniu, Lopes and Babuska, 2012)，确定性政策梯度方法 (Silver 等，

2014 年)，off-policy gradient methods (Degris, White, and Sutton, 2012); 以及熵的正则化 (参见舒尔曼、陈和阿贝尔，2017)。主要应用包括特技直升机自动转盘和 AlphaGo(第 16.6 节)。我们在本章的介绍主要基于 Sutton、McAllester、Singh 和 Mansour(2000)，他们引入了术语“政策梯度方法”。Bhatnagar 等人 (2009) 提供了一个有用的概述。最早的相关作品之一是 Aleksandrov, Sysoyev 和 Shemeneva(1968)。托马斯 (2014) 第一次意识到 t 的因素, 按照本章的装箱算法, 需要打折的情景性问题。

13.1 示例 13.1 和本章的结果与 Eric 一起开发坟墓。

13.2 这里和 334 页上的政策梯度定理是由 Marbach 首先得到的 tsiklis(1998, 2001) 和 Sutton 等人 (2000)。Cao 和 Chen(1997) 也得到了类似的表达。其他早期的结果是由 Konda 和 tsiklis(2000, 2003)、Baxter 和 Bartlett(2001)、Baxter、Bartlett 和 Weaver(2001) 得出的。Sutton, Singh 和 McAllester(2000) 开发了一些额外的结果。

13.3 加强是威廉姆斯的功劳 (1987,1992)。Phansalkar 和 Thathachar (1995) 证明了改进的增强算法的局部和全局收敛定理。全行为算法首先出现在一篇未发表但广为流传的不完整论文中 (Sutton, Singh，和 McAllester, 2000)，然后 Asadi, Allen, Roderick, Mohamed, Konidaris，和 Littman(2017) 对其进行了分析和实证研究，他们称之为“刻薄演员批评家”算法。Ciosek 和 Whiteson(2018) 开发了对持续行动的扩展，他们称之为“预期政策梯度”。

13.4 基线是在 Williams(1987, 1992) 的原始工作中引入的。Greensmith, Bartlett 和 Baxter(2004) 分析了一个可能更好的基线 (参见 Dick, 2015)。Thomas 和 Brunskill(2017) 认为，一个依赖于行动的基线可以在不产生偏见的情况下使用。

13.5-6 的评价方法是最早被调查的强化方法之一。学习 (威滕,1977; 巴托，萨顿，安德森，1983 年; 萨顿,1984)。这里介绍的算法基于 Degris、White 和 Sutton(2012) 的工作。在文献中，演员-批评家方法有时被称为优势-演员-批评家方法。

第一个展示如何以这种方式处理连续操作的例子出现了成为威廉姆斯 (1987,1992)。第 335 页上的数字来自维基百科。

Part III

前沿的研究

在这本书的最后一部分中，我们超越了书的前两部分中提出的标准强化学习思想，简要地考察了它们与心理学和神经科学的关系，强化学习应用的样本，以及未来强化学习研究的一些活跃前沿。

13.10 第 14 章心理学

在前几章中，我们单独基于计算考虑开发了算法的思想。在这一章中，我们从另一个角度来看待这些算法: 心理学的视角以及对动物如何学习的研究。本章的目标是，首先，讨论强化学习思想和算法与心理学家发现的动物学习相关的方法，其次，解释强化学习对学习动物学习的影响。强化学习系统化任务、返回和算法所提供的清晰形式证明对理解实验数据、提出新类型的实验以及指出可能对操作和测量至关重要的因素非常有用。长期优化回报的想法是强化学习的核心，它有助于我们理解动物学习和行为的其他令人困惑的特征。强化学习与心理学理论之间的一些对应关系并不令人惊讶，因为强化学习的发展受到了心理学学习理论的启发。然而，正如本书所阐述的，强化学习从人工智能研究者或工程师的角度探索理想化的情况，目的是用高效的算法解决计算问题，而不是复制或详细解释动物如何学习。因此，我们所描述的一些通信联系了各自领域中独立产生的思想。我们相信这些接触点是特别有意义的，因为它们揭示了对学习很重要的计算原理，无论是人工学习还是自然系统学习。在很大程度上，我们描述了强化学习和学习理论之间的对应关系，这些理论用来解释像老鼠、鸽子和兔子这样的动物是如何在受控的实验室实验中学习的。这些实验在整个 *20* 世纪进行了数千次，其中许多至今仍在进行。尽管这些实验有时被认为与心理学中更广泛的问题无关，但它们探究了动物学习的微妙特性，而这些特性往往是由精确的理论问题所激发的。随着心理学将注意力转移到行为的认知方面，也就是思维和推理等心理过程，动物学习实验开始发挥作用

他们在心理学上的作用比以前小了。但是这种实验导致了学习原理的发现，这些原则在动物王国中是基本且普遍的，在设计人工学习系统时不应忽视这些原则。此外，正如我们将看到的，认知处理的某些方面自然地与强化学习提供的计算视角相连。本章的最后一节包含了与我们讨论的连接以及我们忽略的连接相关的引用。我们希望本章鼓励读者更深入地探究所有这些联系。最后一节还讨论了强化学习中使用的术语与心理学术语之间的关系。强化学习中使用的许多术语和短语都是从动物学习理论中借来的，但是这些术语和短语的计算/工程含义并不总是与它们在心理学中的含义一致。

13.11 预测和控制

我们在本书中描述的算法分为两大类: 预测算法和控制算法。在第三章提出的强化学习问题的解决方法中，这些类别自然产生。在许多方面，这些类别分别对应于心理学家广泛研究的学习类别: 古典或巴甫洛夫，条件作用和工具性，或操作性，条件作用。这些对应关系并不是完全偶然的，因为心理学对强化学习的影响，但它们仍然是惊人的，因为它们把来自不同目标的想法联系在一起。在这本书中提出的预测算法估计数量，这取决于一个代理环境的特征在未来会如何展开。我们特别关注的是评估一个代理在与环境交互时在未来可能收到的回报的数量。在这个角色中，预测算法是策略评估算法，是改进策略的算法的组成部分。但预测算法并不局限于预测未来的回报; 他们可以预测环境的任何特征 (比如，Modayil, White，和 Sutton, 2014)。预测算法和经典条件作用之间的对应关系建立在预测即将到来的刺激的共同属性上，无论这些刺激是否有益 (或有害)。在工具性或操作性条件反射实验中，情况是不同的。在这里，实验装置被设置好，这样动物就会得到它喜欢的东西 (奖励) 或它不喜欢的东西 (惩罚)，这取决于动物做了什么。动物学会增加产生奖励行为的倾向，减少产生惩罚行为的倾向。强化刺激据说取决于动物的行为，而在经典条件作用下则不是 (尽管在经典条件作用实验中很难去除所有行为的偶然性)。工具条件作用实验就像那些启发了桑代克效应定律的实验

控制对我们的意义不同于动物学习理论的意义; 在那里，环境控制着代理，而不是反过来。请看我们在本章末尾对术语的评论。

我们将在第一章中简要讨论。控制是这种学习形式的核心，它对应于强化学习策略改进算法的操作。从预测的角度考虑经典条件作用，从控制的角度考虑工具条件作用，是我们将强化学习的计算观与动物学习联系起来的起点，但现实情况比这更为复杂。经典条件反射比预测更重要; 它也包括行动，还有一种控制模式，有时被称为巴甫洛夫控制。此外，经典的和工具性的条件作用以有趣的方式相互作用，这两种学习方式都可能参与到大多数实验情境中。尽管有这些复杂性，将经典/工具区分与预测/控制区分对齐是将强化学习与动物学习联系起来

的一种方便的第一近似。在心理学中，强化一词被用来描述古典条件作用和工具条件作用下的学习。它最初只指行为模式的强化，也经常用于弱化行为模式。被认为是行为变化原因的刺激被称为强化物，不管它是否取决于动物先前的行为。在本章末尾，我们将更详细地讨论这个术语，以及它与机器学习中使用的术语之间的关系。

13.12 经典条件作用

俄罗斯著名生理学家伊凡 • 巴甫洛夫在研究消化系统的活动时发现，动物对某些触发刺激的先天反应可能是由与先天诱因完全无关的其他刺激触发的。他的实验对象是那些接受过小手术的狗，以便精确测量它们唾液反射的强度。在他描述的一个案例中，这只狗在大多数情况下都不会流口水，但在喂食 5 秒后，它会在接下来的几秒内分泌出 6 滴唾液。在多次重复呈现与食物无关的刺激之后，在这个例子中是节拍器的声音，在食物引入前不久，狗会对节拍器的声音产生唾液反应，就像它对食物的反应一样。“因此，唾液腺的活动被声音的冲动所激活——一种与食物完全不同的刺激”(巴甫洛夫，1927，第 22 页)。巴甫洛夫总结了这一发现的意义，写道:

很明显，在自然条件下，正常的动物不仅必须对刺激做出反应，这些刺激本身就能带来直接的好处或伤害，而且还必须对其他物理或化学介质——声音、光和类似的波——做出反应，而这本身只能表明这些刺激的接近; 虽然它不是野兽的视觉和声音，它本身对较小的动物是有害的，但它的牙齿和爪子。(巴甫洛夫,1927 年,p . 14)

以这种方式将新刺激与先天反射联系起来，现在被称为经典条件作用。巴甫洛夫 (或者更确切地说，他的译者) 称之为“天生反应”(如上图所示的唾液分泌)“无条件反应”(URs)，他们的。

自然触发刺激 (如食物)“无条件刺激”(USs)，以及由预测刺激 (如唾液分泌) 触发的新反应“条件反应”(CRs)。一种最初是中性的刺激，这意味着它通常不会引起强烈的反应 (例如，节拍器的声音)，成为一种“条件刺激”(CS)，因为动物了解到它预测了美国，所以在回应 CS 时产生了 CR。这些术语仍然用于描述经典条件作用实验 (尽管更好的翻译应该是“有条件的”和“无条件的”，而不是条件的和无条件的)。美国被称为强化物，因为它强化了对 CS 的反应生成 CR。

跟踪调节延时调节 CS 我们

CS

我们三军情报局在两种常见的经典条件反射实验中，刺激物的排列被显示在右侧。在延迟条件作用下，CS 贯穿于刺激间期 (ISI)，即 CS 开始和美国开始之间的时间间隔 (CS 结束时，美国以这里所示的公共版本结束)。在跟踪条件作用中，CS 结束后美国开始，CS 偏移与 US 开始的时间间隔称为跟踪间隔。t 巴甫洛夫的狗对着节拍器的声音流口水只是经典条件作用的一个例子在许多动物的反应系统中进行了深入的研究。在某些方面，URs 往往是预先准备好的，比如巴甫洛夫的狗的唾液分泌，或者以某种方式的保护，比如眼睛对刺激眼睛的事物的反应而眨眼，或者看到捕食者而被冻住。在一系列实验中，动物体验到 CS-US 的预测关系后，会知道 CS 可以预测美国，这样动物就可以用 CR 对 CS 做出反应，为动物做好准备，或者保护它不受预测的美国影响。一些 CRs 与 UR 相似，但开始的时间较早，并且在提高效率的方式上有所不同。例如，在一项深入研究的实验中，一种声音 CS 可以可靠地预测一只兔子的眼睛会吸入一股空气 (美国)，引发一种 UR，这种 UR 包括一个被称为“镍钛膜”的保护性内眼睑的闭合。在一次或多次试验之后，这种音调就会触发一种由膜闭合组成的 CR 反应，这种反应从喷气之前开始，最终会被计时，因此，当可能发生喷气时，就会出现峰值闭合。这种 CR，是在预期的空气膨胀和适当的时间，提供更好的保护，而不是简单地启动关闭作为对激怒我们的反应。通过学习刺激之间的预测关系来预测重要事件的能力是非常有益的，它在整个动物王国中广泛存在。

13.12.1 阻塞和高阶条件作用

实验中观察到许多经典条件作用的有趣性质。除了 CRs 的预期性质外，两种被广泛观察的特性在经典条件反射模型的发展中起着重要作用: 阻塞和高阶条件作用。阻塞发生在一个动物不能学习 CR 时提出了一个潜在的 CS 连同另一个 CS, 一直以前条件的动物生产 CR。例如, 在第一阶段的实验涉及到兔子瞬膜条件反射, 一只兔子是第一条件语气 CS 和 CR 的空气吹我们生产关闭瞬膜的空气泡芙。实验的第二阶段包括额外的实验，在这个实验中，第二种刺激，比如一盏灯，被添加到音调中，形成一个复合的音调/灯光 CS，然后同样的空气吹散我们。在实验的第三阶段, 第二个刺激, 光给兔子兔子是否已经学会应对 CR。事实证明, 兔子生产很少, 甚至没有,CRs 的光: 学习光已经被前面的学习了基调。这样的阻塞结果挑战了条件作用只依赖于简单的时间连续的观点，也就是说，条件作用的一个必要而充分的条件是一个美国经常在时间上紧跟着一个 CS。在下一节中，我们将描述 Rescorla - Wagner 模型 (Rescorla 和 Wagner, 1972)，它为阻塞提供了一个有影响力的解释。高阶条件反射发生在一个先前条件下的 CS 扮演一个美国的角色来调节另一个最初的中性刺激。巴甫洛夫描述了一项实验，实验中他的助手首先让狗对着节拍器的声音垂涎三尺，这个节拍器可以预测我们的食物。在这一阶段的训练之后，进行了一系列的实验，在实验中，狗最初并不在意的一个黑色方块被放置在狗的视线中，接着是节拍器的声音——而这并不是食物。在仅仅十次试验中，这只狗仅仅在看到黑广场时就开始流口水，尽管它的出现并没有被食物所吸引。节拍器本身的声音就像一个美国，把唾液分泌的声音转化成黑色方块 c。这是二阶条件。如果黑色方块被用来作为一个美国建立唾液分泌 CRs 到另一个中立的 CS，它将是三级条件作用，等等。高阶条件作用很难证明，尤其是在高阶条件作用下，部分原因是高阶强化物由于在高阶条件作用试验中没有被原来的美国重复跟随而失去其强化价值。但在适当的条件下，如将一阶试验与高阶试验混合，或通过提供普遍的激励刺激，高阶条件作用可被证明超越二阶。正如我们下面所描述的，经典条件作用的 TD 模型使用了引导思想，这是我们的方法的核心，来扩展 Rescorla-Wagner 模型对阻塞的描述，包括 CRs 的预期性和高阶条件作用。高阶的仪器条件反射也发生了。在这种情况下，刺激

与控制组的比较是必要的，以表明先前对音调的条件作用是阻碍对光线的学习。这是通过音调/灯光 CS 的试验来实现的，但是没有之前的调理。在这种情况下，光的学习是没有损害的。摩尔和 Schmajuk (2008) 对这一程序做一个完整的说明。

一致预测初级强化本身就是一种强化物，如果它的奖励或惩罚的质量是通过进化建立在动物身上的，那么强化就是第一种强化物。预测刺激变成了次级强化物，或者更一般地说，高阶或条件强化物——后者是一个更好的术语当预测的强化刺激本身是次级的，甚至是更高阶的强化物。条件强化物提供条件强化: 条件奖励或条件惩罚。条件性强化就像一级强化一样，增加了动物产生条件性奖励行为的倾向，减少了动物产生条件性惩罚行为的倾向。(请参阅我们在本章末尾的评论，它解释了我们的术语有时是如何与心理学中使用的术语有所不同的。) 条件强化是一个关键的现象，它解释了为什么我们为条件强化金钱工作，它的价值仅仅来自于拥有它所预言的东西。在第 13.5 节所描述的演员-评论家方法中 (在第 15.7 和 15.8 节的神经科学中进行了讨论)，评论家使用 TD 方法来评价演员的政策，其价值估计为演员提供条件强化，允许演员改进其政策。这种高阶工具条件作用的模拟有助于解决第 1.7 节中提到的信贷分配问题，因为当主奖励信号被延迟时，评论家会对参与者进行时时刻刻的强化。我们将在下面的 14.4 节中讨论这个问题。

13.12.2 Rescorla-Wagner 模型

Rescorla 和 Wagner 创建他们的模型主要是为了解释阻塞。Rescorla-Wagner 模型的核心理念是，只有当动物感到意外时，动物才会学习，换句话说，只有当动物感到意外的时候 (尽管没有必要暗示任何有意识的期望或情感)。我们首先用术语和符号表示 Rescorla 和 Wagner 的模型，然后转向我们用来描述 TD 模型的术语和符号。这就是 Rescorla 和 Wagner 对他们的模型的描述。该模型调整了复合 CS 的每个成分刺激的“联想强度”，这个数字表示该成分预

测美国的强度或可靠性。当 CS 复合构成的几个组件刺激呈现在经典条件作用试验中, 每个组件的关联强度刺激变化的方式取决于一个关联强度与整个刺激化合物, 称为“聚合关联强度, 而不只是在每个组件的关联强度本身。Rescorla 和 Wagner 认为这是一种化合物 CS AX，由 a 和 X 组成，动物可能已经经历过 a 刺激，刺激 X 对动物来说可能是新的。让 VA、VX 和 VAX 分别表示刺激 A、X 和化合物 AX 的结合力。假设在试验中，化合物 CS AX 后面跟着一个 US，我们把它命名为刺激物 y，然后是它的结合力

刺激成分变化根据这些表达式:ΔVA = A Y(RY−VAX) ΔVX = X Y(RY−VAX), A Y 和 X Y 步长参数, 取决于 CS 组件和美国的身份, 和联想的渐近水平从总体实力, 美国可以支持。(Rescorla 和瓦格纳在这里 R 的使用, 但我们使用 R 来避免混淆使用 和因为我们通常认为这是一个奖励信号的大小, 与经典条件作用的警告, 美国不一定是奖励或惩罚)。该模型的一个关键假设是，总联合强度 VAX 等于 VA + VX。联想的优势, 改变了这些 Δs 成为联想的优势开始下一个试验。为了完成这个模型，模型需要一个响应生成机制，这是一种将 V 的值映射到 CRs 的方法。因为这个映射依赖于实验情况的细节，Rescorla 和 Wagner 没有指定映射，只是假设更大的 V s 会产生更强或更可能的 CRs，而负的 V s 意味着没有 CRs。Rescorla-Wagner 模型解释了以一种解释阻塞的方式获取 CRs。VAX, 只要总关联强度的刺激化合物低于渐近水平的关联强度„ 我们可以支持, 预测误差的 RY−VAX 是正的。这意味着，在连续的试验中，组成刺激的结合强度 VA 和 VX 会增加，直到总结合强度 VAX 等于 RY，这时，结合强度停止变化 (除非美国发生变化)。当一个新的组件添加到复合 CS 的动物已经条件, 进一步调节增强复合产生很少或没有添加 CS 组件的关联强度增加, 因为错误已经减少到零, 或一个较低的值。美国的出现几乎已经得到了完美的预测，因此新的 CS 组件很少或没有错误 (或令人惊讶)。之前的学习妨碍了对新组件的学习。为了从 Rescorla 和 Wagner 的模型过渡到经典条件作用的 TD 模型 (我们称之为 TD 模型)，我们首先根据我们在整本书中使用的概念重新塑造他们的模型。具体地说, 我们匹配的符号使用学习线性函数近似 (9.4 节), 我们认为学习的调节过程作为一个预测的“我们的”试验的基础上提出的复合 CS 的审判, 美国 Y 的大小在哪里上面给出的 Rescorla-Wagner 模型的变化。我们也介绍。因为 Rescorla-Wagner 模型 trial-level 模型, 这意味着它处理关联优势如何改变从试验, 试验不考虑任何细节和之间的试验中发生了什么, 我们不需要考虑如何在试验状态改变时, 直到我们在以下部分呈现完整的 TD 模型。相反，在这里，我们简单地认为状态是一种标记试验的方式，根据试验中出现的 CSs 组件的集合。因此，假设试验类型或状态 s 由特征的实值向量 x(s) = (x1(s)，x2(s)，⋯ ,xd(s))? 如果 CSi 是 a 的第 i 个分量，xi(s) = 1

复方 CS，在试验中出现，否则为 0。然后，如果联合强度的 d 维矢量为 w，则三型 s 的总联想强度为。

v̂(s,w)= w ? x(s)。(14.1) 这对应于强化学习的价值估计，我们把它看作是美国的预测。现在暂时让 t 表示一个完整的试验的数量, 而不是它通常意义作为一个时间步 (我们回到 t 通常意味着当我们扩展到下面的 TD 模型), 并假定圣是国家相应试验 t。调节试验 t 更新关联强度矢量 wt wt + 1 如下:

wt + 1 = wt + tx(圣), (14.2)

是步长参数,——因为在这里, 我们描述 Rescorla -瓦格纳 model- t 预测误差

t = Rt−v̂(St,wt)。(14.3) Rt 是试验 t 的预测目标，也就是美国的规模，或者用 Rescorla 和 Wagner 的话来说，试验中的美国能支持的结合力。注意，由于 (14.2) 中的因子 x(St)，只有在试验中出现的 CS 组件的结合力作为试验结果进行了调整。你可以把预测误差看作是一种惊喜的度量，而总联想强度则是当动物的期望值与目标美国的大小不匹配时所违反的期望。从机器学习的角度看，Rescorla-Wagner 模型是一个纠错监督学习规则。它本质上和最小均方 (LMS) 或 Widrow-Hoff，学习规则 (Widrow and Hoff, 1960) 一样，发现权值——这里的结合力——使所有误差的平方的平均值尽可能接近于零。它是一种“曲线拟合”或回归算法，在工程和科学应用中得到广泛应用 (见第 9.4 节)。Rescorla-Wagner 模型非常有影响力的动物学习理论的历史, 因为它表明, 一个“机械”的理论可以解释关于阻塞的主要事实不需要采取更复杂的认知理论涉及, 例如, 动物的明确认识到另一个刺激组件被添加, 然后向后扫描其短期记

忆重新评估涉及美国的预测关系。Rescorla-Wagner 模型展示了如何用一种简单的方式调整传统的条件连续理论 (即刺激的时间连续是学习的必要和充分条件) 来解释阻塞 (Moore and Schmajuk, 2008)。Rescorla-Wagner 模型提供了一个简单的关于阻塞的描述和一些古典条件作用的其他特征，但是它不是一个完整的或完美的古典模型

LMS 规则和 Rescorla-Wagner 模型的唯一区别是 LMS the 输入向量 xt 可以拥有任意实数作为组件, 并且至少在最简单的版本的 LMS 状态步长参数 不依赖于输入向量或刺激的身份背景预测的目标。

调节。不同的想法解释了各种观察到的影响，在理解古典条件作用的许多微妙之处方面仍在取得进展。我们接下来将描述的 TD 模型，虽然也不是一个完整或完美的经典条件作用模型，但它扩展了 Rescorla-Wagner 模型，以解决刺激之间的试验内和试验间的时间关系如何影响学习以及如何产生高阶条件作用。

13.13 TD 模型

TD 模型是一个实时模型，与 Rescorla - Wagner 模型不同。Rescorla-Wagner 模型中的一个单步 t 代表了一个完整的条件作用试验。该模型并不适用于在试验进行期间发生的事情或试验之间可能发生的事情的细节。在每次试验中，动物可能会经历不同的刺激，这些刺激在特定的时间发生，并且持续的时间也不同。这些时间关系强烈影响学习。Rescorla-Wagner 模型也不包含高阶条件作用机制，而对于 TD 模型，高阶条件作用是基于 TD 算法的自举思想的自然结果。为了描述 TD 模型，我们从上面的 Rescorla-Wagner 模型的提法开始，但是 t 现在在试验中或试验之间标注时间步数，而不是完整的试验。认为时间 t,t + 1 之间的一个小的时间间隔, 说幅第二, 和认为审判是一个序列的州, 一个与每个时间步, 现在国家在一步 t 代表刺激是如何表示的细节在 t CS 组件, 而不只是一个标签出现在一个审判。事实上，我们可以完全放弃试验的想法。从动物的角度来看，试验只是它与世界互动的持续经验的一个片段。按照我们通常看到的代理与环境相互作用的观点，想象一下，动物经历了无数的状态序列，每个状态序列都由特征向量 x(s) 表示。也就是说，把试验称为刺激模式在实验中重复的时间片段通常还是比较方便的。状态特征不局限于描述动物所经历的外部刺激; 它们可以描述外部刺激在动物大脑中产生的神经活动模式，这些模式可以依赖于历史，这意味着它们可以是外部刺激序列产生的持久模式。当然，我们并不确切地知道这些神经活动模式是什么，但是像 TD 模型这样的实时模型可以让我们探索学习关于外部刺激的内部表征的不同假设的后果。由于这些原因，TD 模型不提交任何特定的状态表示。此外，由于 TD 模型包含了在刺激之间跨越时间间隔的折现和资格跟踪，因此该模型还可以探索折扣和资格跟踪如何与刺激表示交互，从而对经典条件反射实验的结果进行预测。下面我们将描述一些与 TD 模型一起使用的状态表示，以及它们的一些含义，但是目前我们仍然不知道

表示并假设每个状态 s 都由特征向量 x(s) = (x1(s)，x2(s)，⋯,xn)?。然后，状态 s 对应的总结合力由 (14.1) 给出，与 Rescorla-Wgner 模型相同，但 TD 模型更新的结合力矢量 w 不同。现在 t 标记了一个时间步骤而不是一个完整的试验，TD 模型根据这个更新来支配学习:

wt + 1 = wt + tzt, (14.4) Rescorla-Wagner 取代 xt(St) 的更新和 zt 型 (14.2), 一个向量的资格痕迹, 和 (14.3), 而不是 t t 是 TD 错误: t = Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt), (14.5) 是折现系数 (在 0 和 1 之间), 预测目标在时间 t Rt,v̂(wt 圣 + 1) 和 v̂(St,wt) 聚合关联优势在 t + 1 和 t(14.1) 所定义的。每个组件我 eligibility-trace 矢量 zt 型增量或根据组件的精神性 xi(St) 的特征向量 x(St), 和其他衰变率由 :

zt 型 + 1 = zt + x(St)。(14.6) 是通常的资格跟踪衰减参数。注意, 如果 = 0,TD 模型减少 Rescorla-Wagner 模型的异常:

14.2.4 TD 模型模拟

像 TD 模型这样的实时调节模型之所以有趣，主要是因为它们可以预测许多无法用试验级模型表示的情况。这些情况包括条件刺激的时间和持续时间，这些刺激的时间与美国的时间

有关，以及 CRs 的时间和形状。例如，美国通常必须在一个中性刺激开始后开始进行条件反射，学习的速度和有效性取决于刺激间间隔 (ISI)，即 CS 和美国的一组之间的间隔。当 CRs 出现时，它们通常在美国出现之前就开始了，在学习过程中它们的时间分布也会发生变化。在使用复合 CSs 的条件作用下，复合 CSs 的组件刺激可能不会同时开始和结束，有时会形成所谓的串行化合物，其中的组件刺激会随着时间的推移而发生。像这样的时间考虑使得重要的是考虑刺激是如何表示的，这些表征是如何在试验期间和试验之间随时间展开的，以及它们是如何与折扣和资格痕迹交互的。

图 14.1: 三个刺激表示 (列) 有时用于 TD 模型。每一行表示刺激表示的一个元素。三个代表不同沿时间概化梯度，在全序列化合物 (左列) 中相邻时间点之间不进行概化，在相邻时间之间完全概化存在表示中的点 (右列)。microstimulus 表示占据一个中间立场。时间的泛化程度决定了时间的粒度。通过这些，我们学会了预测。适应学习和行为的微小变化，评价经典条件作用的 TD 模型，第 40 卷，2012，第 311 页，E. A. Ludvig, R. S。萨顿,e . j . 凯赫。施普林格的许可。

图 14.1 显示了用于探究 TD 模型行为的三个刺激表示: 完整的串行化合物 (CSC)、微刺激 (MS) 和存在表示 (Ludvig、Sutton 和 Kehoe, 2012)。这些表示形式的不同之处在于，它们在刺激出现的邻近时间点之间强制泛化的程度不同。图 14.1 中最简单的表示形式是图右列中的表示形式。此表示对于在试验中出现的每个组件 CS 都有一个单独的特性，其中该特性在该组件出现时的值为 1，否则为 0。在场表征并不是关于刺激物如何在动物的大脑中被表现出来的现实假设，但是正如我们下面所描述的，带有这种表征的 TD 模型可以产生许多经典条件作用下看到的时间现象。对于 CSC 表示 (图 14.1 的左列)，每个外部刺激的出现都会引发一系列精确定时的短持续内部信号

4 在我们的形式, 有不同的状态, 圣, 为每个时间步 t 庭审中翻供, 以及试验中, 一个复合 CS 由 n 组件 CSs 各种时间发生在整个试验中, 在不同时期有一个特性,xi, 对于每个组件 CSi,i = 1,。其中 xi(St) = 1 表示当 CSi 存在时，所有的时间都是 t，否则等于 0。

持续到外部刺激结束。这就好比假设动物的神经系统有一个时钟，它可以在刺激时精确地记录时间; 这就是工程师们所谓的“抽头延迟线”。与在场表征一样，CSC 表征作为大脑内部如何表示刺激的假设是不现实的，但 Ludvig 等人 (2012) 称其为“有用的虚构”，因为它可以揭示 TD 模型在相对不受刺激表征约束的情况下是如何工作的细节。CSC 表示也被用在大脑中产生多巴胺的大多数 TD 模型中，这是我们在第 15 章中讨论的话题。CSC 表示经常被视为 TD 模型的一个重要部分，尽管这种观点是错误的。MS 表示法 (图 14.1 的中心列) 类似于 CSC 表示法，因为每个外部刺激都引发一系列内部刺激，但在这种情况下，内部刺激——微刺激——并不是如此有限和不重叠的形式; 它们会随着时间和重叠而延长。随着时间从刺激开始，不同的微刺激组变得或多或少的活跃，每一个后续的微刺激逐渐扩大，并达到一个较低的最高水平。当然, 有很多女士表示根据 microstimuli 的性质, 和很多女士的例子表示研究文献中, 在某些情况下还有建议动物的大脑如何产生 (见文献和历史评论这一章结束时)。MS 表示法比存在或 CSC 表示法更现实，它们是关于刺激物的神经表示法的假设，它们允许 TD 模型的行为与动物实验中观察到的更广泛的现象集合相关。特别是 microstimuli 的假设瀑布是由航空母舰以及 CSs, 并通过研究显著影响学习 microstimuli 之间的交互, 资格痕迹, 打折,TD 模型帮助框架假设占经典条件作用的许多微妙的现象和动物的大脑如何产生。我们将在下面讨论更多，特别是在第 15 章，我们将讨论强化学习和神经科学。然而，即使使用简单的存在表示，TD 模型也产生了 Rescorla-Wagner 模型解释的所有经典条件作用的基本属性，以及超出试验级模型范围的条件作用的特性。例如，正如我们已经提到的，经典条件作用的一个显著特征是，美国通常必须在条件作用出现的中性刺激开始后开始，条件作用发生后，CR 开始于美国出现之前。换句话说，条件反射通常需要一个积极的 ISI，而 CR 通常会预期到美国。条件作用的强度 (例如，CS 引起的 CRs 百分比) 取决于 ISI 在不同物种和反应系统中的差异，但它通常具有以下特性: 对于零 ISI 或负 ISI，它是可以忽略的，即。，当美国发病与 CS 发病同时发生，或比 CS 发病更早时 (尽管研究发现，联想强度有时会略有增加，或与 CS 发生负相关

5 . 在我们的形式主义中，对于每一个 CS 组件的 CSi，在一个试验中，并且在每一个时间 t 在 a。试一下，有一个单独的特性，如果 t = t，那么 xt i(St?) = 1 ? 对于任何 t ? 现场 CSi 和否则等于 0。这与 Sutton 和 Barto(1990) 中的 CSC 表示不同每个时间步骤都有相同的特点，但没有对外界刺激的参考; 因此, 完整的系列化合物名称。

前在第一个 CS (CSA) 和美国之间的空跟踪间隔中填充第二个 CS (CSB)，形成一个串行复合刺激，然后方便条件作用于 CSA。右边显示的是 TD 模型的行为，该模型在模拟这样一个实验中的存在表示，其时间细节如上所示。与实验结果相一致 (Kehoe, 1982)，该模型显示了第一个 CS 在第二次 CS 出现时的调节速率和第一个 CS 的渐近水平。这是一个著名的例子。

圣 e 年代的

道明模型中的埃格-米勒效应在一个试验中，对刺激之间的时间关系的调节作用是由埃格和米勒 (1962) 所做的一个实验，该实验涉及两个重叠的 CSs 在延迟配置中的作用，如图右上方所示。尽管 CSB 与美国的时间关系更好，但 CSA 的存在大大降低了 CSB 与 CSA 不存在的对照组的条件反射。在这个实验的模拟中，直接向右显示了 TD 模型产生的相同结果。TD 模型解释了阻塞，因为它是一个错误校正的学习规则，就像 Rescorla-Wagner 模型一样。除了计算基本的阻塞再

354 年第 14 章: 心理学然而，sults, TD 模型预测 (包括存在表示和更复杂的表示)，如果阻塞的刺激被提前移动，那么阻塞就会被逆转，从而在阻塞发生之前发生

图 14.2:TD 模型中的时间优先级覆盖阻塞。阻断刺激的开始 (如右图中的 CSA)。TD 模型行为的这一特征值得关注，因为在模型引入时没有观察到它。回想一下，在闭锁过程中，如果一个动物已经知道一个 CS 可以预测一个美国人，那么这个新增加的第二个 CS 也可以预测美国人的减少。, 被阻塞。但是，如果新添加的第二个 CS 比预先训练的 CS 开始得早，那么——根据 TD 模型——对新添加 CS 的学习不会被阻塞。事实上，随着训练的继续，新增加的 CS 增加了联想力，训练前的 CS 失去了联想力。在这些条件下，TD 模型的行为如图 14.2 的下半部分所示。这个模拟实验不同于 Egger-Miller 实验 (上一页的底部)，即较短的 CS 在与美国完全相关之前进行了预先训练。这一惊人的预测使得 Kehoe、Schreurs 和 Graham(1987) 使用经过充分研究的兔子显微膜制剂进行了实验。他们的研究结果证实了该模型的预测，他们指出，非 td 模型在解释其数据时存在相当大的困难。与 TD 模型, 早期预测刺激优先于后预测刺激, 因为在这本书中描述的所有预测方法、TD 模型基于倒车或引导理念: 更新关联优势优势在特定状态转向的力量之后。自举的另一个结果是，TD 模型提供了高阶条件作用的描述，这是古典条件作用的一个特性，超出了 Rescoral-Wagner 和类似模型的范围。正如我们上面所描述的，高阶条件反射是一种现象，在这种现象中，先前条件反射的 CS 可以作为一个美国来调节另一个最初的中性刺激。图 14.3 显示了 TD 模型 (同样是存在表示) 在一个高阶条件反射实验中的行为——在这种情况下，它是二级条件反射。在第一阶段 (图中没有显示)，CSB 被训练去预测一个美国，使它的结合力增加，这里是 1.65。在第二阶段，CSA 在没有美国的情况下与 CSB 配对，按照图顶部所示的顺序排列。尽管 CSA 从未与美国配对，但它仍具有结合力。通过持续的训练，CSA 的结合力达到峰值，然后下降，因为第二强化 CSB 的结合力降低，失去了提供二次强化的能力。CSB 的联想

图 14.3:TD 模型的二阶条件反射。因为在这些高阶条件作用试验中，美国没有出现，所以强度会降低。这些是 CSB 的灭绝试验，因为它与美国的预测关系被破坏了，所以它作为强化物的能力下降了。在动物实验中也可以看到同样的模式。在高阶条件反射试验中，条件强化的消失使得演示高阶条件反射变得困难，除非原始的预测关系周期性地通过插入一阶试验来恢复。TD 模型产生第二和高阶条件的模拟, 因为 $\hat{v}$(wt) 圣 + 1−$\hat{v}$(St,wt) 出现在 TD 错误 t(14.5)。这意味着, 由于之前的学习, $\hat{v}$(wt 圣 + 1) 可以不同于 $\hat{v}$(St,wt), 使 t 零 (时间不同)。这种差异与 (14.5) 中 Rt+1 的状态相同，这意味着，在学习方面，时间差异和发生在美国之间没有区别。实际上，TD 算法的这一特性是其发展的主要原因之一，我们现在通过它与动态编程的联系理解了这一点，如第 6 章所述。自举值与二阶、高阶条件作用密切相关。

在上面描述的 TD 模型行为的例子中，我们只研究了 CS 组件的关联强度的变化; 我们没

有研究模型对动物条件反射 (CRs) 特性的预测: 它们的时间、形状，以及它们在条件反射试验中是如何发展的。这些特性取决于物种、观察到的反应系统和条件作用试验的参数，但在许多不同动物和不同反应系统的实验中，CR 的大小或 CR 的概率会随着美国的预期时间的增加而增加。例如, 在经典条件作用的一只兔子的瞬膜反应, 我们上面提到的, 在条件反射实验中延迟从 CS 出现当瞬膜开始跨越眼随试验, 和这个先行关闭的振幅逐渐增加在 CS 和美国之间的间隔, 直至膜达到最大关闭预计美国时间。CR 的时间和形状对其适应性意义至关重要——过早地覆盖眼睛会降低视力 (即使硅胶膜是半透明的)，而过晚覆盖则没有什么保护价值。对于经典条件作用的模型来说，获取这样的 CR 特征是一个挑战。TD 模型不包括其定义任何机制来翻译我们预测的时间进程,v̂(St,wt), 到一个配置文件, 可以与动物的 CR。最简单的属性选择的时间进程

模拟的 CR 等于美国预测的时间过程。在这种情况下, 模拟 CRs 的特点以及它们如何改变试验只取决于刺激表示选择和模型的参数的值 、 , 。图 14.4 显示了我们在学习过程中不同时间点的预测时间过程，如图 14.1 所示。这些模拟美国发生 25 次步骤发病后的 CS, = .05, = .95 和 = .97 点。通过 CSC 表示 (图 14.4 左)，TD 模型形成的美国预测曲线在 CS 和美国之间的区间内呈指数增长，直到达到美国发生时的最大值 (在第 25 步)。这种指数增长是 TD 模型学习规则折现的结果。由于存在表示法 (图 14.4 中)，在刺激出现时，美国的预测几乎是不变的，因为只有一种重量，或联想强度，用于每一种刺激的学习。因此，具有存在表示的 TD 模型不能重新创建 CR 计时的许多特性。使用 MS 表示 (图 14.4 右侧)，TD 模型的美国预测的开发就更加复杂了。经过 200 次试验，预测的轮廓是用 CSC 表示生成的美国预测曲线的合理近似。

图 14.4: 三种不同刺激表征的 TD 模型在获取过程中的时间过程预测。左: 有了完整的串行化合物 (CSC)，美国的预测在时间间隔中呈指数增长，在美国达到顶峰。在渐近线 (试验 200)，美国的预测在美国强度上达到顶峰 (在这些模拟中为 1)。中间: 在存在表示的情况下，美国的预测几乎是不变的。的水平。这个恒定的水平是由美国强度和 CS-US 间隔的长度决定的。右: 微刺激表示，在渐近点，TD 模型接近用 CSC 通过线性组合表示的指数增长的时间过程 microstimuli 不同。适应学习和行为的微小变化，评价经典条件作用的 TD 模型，第 40 卷，2012,E. A. Ludvig, R. S. Sutton, E. J. Kehoe。施普林格的许可。

图 14.4 所示的美国预测曲线并不是为了精确地匹配在任何特定动物实验条件作用下的 CRs 曲线，但它们说明了刺激表现对从 TD 模型得出的预测的强大影响。此外，尽管我们只能在这里提到，刺激方案如何与折扣和资格跟踪相互作用，这一点很重要。

在确定由 TD 模型产生的美国预测剖面的性质时。另一个超越我们在这里讨论的维度是不同的反应生成机制的影响，这些机制将我们的预测转化为 CR 档案; 图 14.4 所示的剖面图是“原始”的美国预测剖面图。即使没有任何特殊的假设关于动物的大脑可能会产生明显的反应我们的预测, 然而,CSC 的概要文件在图 14.4 和女士表示增加的时间我们方法和达到最大时的我们, 就像在许多动物条件反射实验。TD 模型，结合特定的刺激表示和反应生成机制，能够解释在动物经典条件反射实验中观察到的异常广泛的现象，但它远不是一个完美的模型。为了生成经典条件作用的其他细节，需要对模型进行扩展，可能需要添加基于模型的元素和机制，以便自适应地修改其某些参数。其他建模经典条件反射的方法明显地偏离了 rescorla - wagner 风格的错误修正过程。例如，贝叶斯模型在经验修正概率估计的概率框架内工作。所有这些模型都有助于我们理解经典条件作用。TD 模型的最显著特点是它是基于一个理论的理论我们已经描述了在这个书, 表明的是什么动物的神经系统要做而进行调节: 它试图形成准确的长期预测, 符合所强加的限制表示刺激的方式以及神经系统是如何运作的。换句话说，它提出了一个经典条件作用的规范性解释，在这个解释中，长期而非直接的预测是一个关键特征。经典条件作用的 TD 模型的发展就是一个明确的目标是对动物学习行为的一些细节进行建模的实例。除了作为一种算法的地位，TD 学习也是生物学习这一模型的基础。正如我们在第 15 章中所讨论的，TD 学习也证实了一个有影响的模型，即产生多巴胺的神经元的活动。多巴胺是哺乳动物大脑中的一种化学物质，与奖赏过程密切相关。在这些例子中，强化学习理论与动物行为和神经数据进行了详细的联系。我们现在转向考虑工具条件作用实验中强化

学习和动物行为之间的对应关系，这是动物学习心理学家研究的另一种主要的实验类型。

13.14 工具性条件作用

在工具性条件反射实验中，学习依赖于 be-havior 的后果: 一种强化刺激的传递取决于动物的行为。相比之下，在经典的条件反射实验中，强化刺激——美国——是独立于动物行为的。器质性条件作用通常被认为和操作性条件作用是一样的，b • f • 斯金纳 (1938,1963) 在行为-或有强化实验中引入的术语

使用这两个术语的人的观点和理论有很多不同之处，其中一些我们将在下文中提及。我们将专门用“工具条件作用”这个词来描述实验，在实验中，强化取决于行为。工具性条件作用的根源可以追溯到美国心理学家爱德华 • 桑代克 (Edward Thorndike) 所做的实验。桑代克观察了猫的行为。

桑代克的益智盒之一。《动物智力: 动物联想过程的实验研究》,《心理学评论》，第二 (4) 专著系列，纽约麦克米伦出版社，1898 年版。当他们被放置在“拼图盒子”中，例如右边的那个，他们可以通过适当的行动逃跑。例如，一只猫可以通过三个不同的动作来打开一个盒子的门: 把盒子后面的一个平台压下去，抓着一根绳子拉绳子，把一根棍子向上或向下推。当它们第一次被放在一个益智盒里，外面可以看到食物时，除了桑代克的一些猫外，其他的猫都表现出“明显的不适迹象”和异常剧烈的活动，“本能地想要逃离监禁”(桑代克，1898)。用不同的猫和有不同逃生机制的盒子里，桑代克记录了每只猫在每个盒子里经历多次逃生所花费的时间。他观察到时间几乎总是随着连续的经历而减少，例如，从 300 秒到 6 秒或 7 秒。他在一个益智盒里这样描述猫的行为:

那只猫在她的冲动挣扎中在盒子上抓来抓去的猫很可能会用爪子抓绳子、圈或按钮来开门。渐渐地，所有其他的不成功的冲动都会被消除，而导致成功行为的特定冲动会被由此带来的快乐所压制，直到，经过多次尝试，猫在被放进盒子后，会立即以一种确定的方式抓住按钮或循环。(桑代克 1898 年,p . 13) 这些实验和其他的实验 (有些是用狗、小鸡、猴子甚至鱼做的) 让桑代克制定了一些学习的“法则”，其中最具影响力的是“效果法则”，我们在第 1 章 (第 15 页) 中引用了它的一个版本。这条定律描述了人们通常所说的“试错法”。正如第一章中所提到的，影响定律的许多方面都引起了争议，其细节也在多年来不断修改。然而，法律——无论哪种形式——仍然表达了一种持久的学习原则。强化学习算法的本质特征与动物学习的特征相对应。首先，强化学习算法是可选择的，这意味着他们会尝试不同的选择，并通过比较它们的结果进行选择。第二，强化学习算法是关联的，这意味着通过选择找到的替代方法与特定的情况或状态相关联，从而形成代理策略。就像学习用效应定律来描述，强化

学习不仅是寻找能产生大量回报的行为的过程，而且是将这些行为与情景或状态联系起来的过程。桑代克通过“选择和连接”(Hilgard, 1956) 使用了这个短语学习。进化中的自然选择是选择过程的一个主要例子，但它不是关联的 (至少正如人们普遍理解的那样); 监督学习是有关联的，但它不是选择性的，因为它依赖指令直接告诉代理如何改变它的行为。在计算术语中，效果定律描述了一种将搜索和记忆结合在一起的基本方法: 在每种情况下尝试和选择许多操作的形式中进行搜索，在将情况与发现的操作联系起来的形式中进行记忆——到目前为止——在这些情况下效果最佳。搜索和记忆是所有强化学习算法的基本组成部分，无论记忆是采用代理策略、值函数还是环境模型的形式。强化学习算法的搜索需求意味着它必须以某种方式进行探索。动物们显然也在探索，早期的动物学习研究人员对一种动物在类似桑代克的益智盒的情况下选择其行为的指导程度并不认同。行为是“绝对随机、盲目摸索”的结果 (伍德沃斯，1938 年，第 777 页)，还是有某种程度的指导，要么是通过事先学习、推理，要么是其他方式? 尽管包括桑代克在内的一些思想家似乎已经采取了前者的立场，但其他人则倾向于更深思熟虑的探索。增强学习算法为代理在选择行为时可以使用多少指导提供了很大的自由度。形式的探索我们使用在这本书中给出的算法, 如 -greedy 和 upper-confidence-bound 行动选择, 仅仅是最简单的。更复杂的方法是可能的，唯一的规定是必须对算法进行某种形式

的探索才能有效地工作。我们处理强化学习的特性允许一组行动在任何时候都依赖于环境的当前状态，这与 Thorndike 在他的猫的益智盒行为中观察到的一些东西相呼应。这些猫从它们本能地在当前环境下的行为中选择动作，桑戴克称之为它们的“本能冲动”。首先，猫被放在一个益智盒里，它本能地用巨大的能量抓挠、抓爪和咬东西: 猫在有限的空间中找到自己时的本能反应。成功的行动是从这些行动中选择的，而不是从每个可能的行动或活动中选择。这就像我们形式主义的特征，从一个国家中选择的行为属于一组可接受的行为，a (s)。指定这些集合是增强学习的一个重要方面，因为它可以从根本上简化学习。它们就像动物的本能冲动。另一方面，桑代克的猫可能是根据本能的特定情境而不是行动的顺序来探索，而不是仅仅从一组本能的冲动中进行选择。这是另一种使强化学习更容易的方法。受影响的最著名的动物学习研究者包括克拉克 • 赫尔 (如赫尔，1943) 和 b • f • 斯金纳 (如斯金纳，1938)。他们研究的中心是基于行为的后果来选择行为。强化学习具有与赫尔理论相同的特点，包括类似于精英的机制和二次强化，以解释当行为与随后的强化之间存在显著的时间间隔时的学习能力

刺激 (见 14.4 节)。随机性也在赫尔的理论中发挥了作用，通过他所谓的“行为振荡”来引入探索性行为。斯金纳并没有完全认同记忆方面的影响定律。他反对联想联系的概念，相反，他强调从自发的行为中选择。他引入了“操作性”一词来强调行动对动物环境的影响的关键作用。与桑代克和其他实验不同的是，斯金纳的操作条件作用实验允许动物实验在长时间内不间断地进行。他发明了操作性条件反射室，现在被称为“斯金纳箱”，其中最基本的版本包含一个杠杆或钥匙，动物可以按这个杠杆或钥匙来获得奖励，比如食物或水，这些奖励将根据一个定义明确的规则 (称为强化计划) 交付。通过记录杠杆按压的累积次数作为时间的函数，斯金纳和他的追随者可以研究不同的强化时间表对动物杠杆按压速度的影响。使用我们在本书中介绍的强化学习原理来模拟这些实验的结果并不是很完善，但是我们在本章末尾的书目和历史评论部分中提到了一些例外。斯金纳的另一项贡献来自于他对训练动物的有效性的认识，他通过不断强化对所期望行为的近似，他称之为“塑造”过程。虽然这种方法已经被包括斯金纳在内的其他人使用过，但是当他和他的同事们试图用鸽子的嘴敲击一个木球来训练鸽子打碗的时候，这种方法的重要性却给他留下了深刻的印象。在等待了很长一段时间后，他们没有看到任何可以加强的迹象…决定要加强任何与“swipea”有细微相似之处的反应——也许，起初仅仅是观察球的行为，然后选择更接近最终形式的反应。结果让我们感到吃惊。不出几分钟，球就从箱子的墙壁上滚下来，好像鸽子是壁球冠军似的。(斯金纳,1958,p. 94) 鸽子不仅学会了一种对鸽子来说不寻常的行为，它还通过一种相互作用的过程迅速地学会了这种行为，在这个过程中，鸽子的行为和强化事件会因彼此的反应而发生变化。斯金纳将加固过程与雕刻家将黏土塑造成理想的形状进行了比较。整形是计算强化学习系统的一种强有力的技术。当一个代理很难收到任何非零的奖励信号时，无论是由于奖励情况的稀疏性，还是由于初始行为的不可接近性，从一个更容易的问题开始，随着代理学习逐渐增加难度，都可以是一种有效的、有时是不可缺少的策略。来自心理学的一个概念在工具条件作用的背景下特别相关，那就是动机，它是指影响行为的方向和力量或活力的过程。例如，桑代克的猫被驱使着逃离益智盒，因为它们想要的是外面的食物。获得这个目标对他们是有益的，并加强了允许他们逃跑的行动。很难把动机的概念精确地联系起来，因为它有很多维度

强化学习的计算视角，但与它的一些维度有明显的联系。从某种意义上说，强化学习行为者的奖励信号是其动机的基础: 行为者的动机是使其长期获得的总奖励最大化。因此，动机的一个关键方面是，是什么让一个代理人的经验值得。在强化学习中，奖励信号取决于强化学习主体的环境和行为状态。此外，正如第 1 章所指出的，代理环境的状态不仅包括关于机器外部的信息，如存放代理的有机体或机器人，还包括机器内部的信息。一些内部状态成分对应于心理学家所说的动物的动机状态，它影响着对动物的奖励。例如，动物在饥饿的时候吃比刚吃完一顿令人满意的饭时吃得更多。国家依赖的概念足够广泛，可以允许多种类型的调制对奖励信号产生的影响。价值函数为心理学家的动机概念提供了进一步的联系。如果选择一个行动的最基本的动机是为了获得尽可能多的回报, 为强化学习代理选择操作使用价值函数, 一个更近的动机是提升其价值函数的梯度, 即选择行为将导致大多数高估值的下一个国家 (或本质上是一样的, 选择行动最大的 action-values)。对于这些代理，价值函数是决定其行为

方向的主要驱动力。动机的另一个维度是，动物的动机状态不仅影响学习，而且也影响动物学习后行为的强度或活力。例如，在学会在迷宫的目标盒子里找到食物后，饥饿的老鼠比不饿的老鼠跑得更快。动机的这个方面并没有如此清晰地与我们在这里展示的强化学习框架联系起来，但是在这一章末尾的书目和历史评论部分，我们引用了几篇论文，提出了基于强化学习的行为活力理论。当强化刺激在强化事件后发生时，我们现在转向学习的主题。强化学习算法用于延迟强化学习的机制——合格跟踪和 TD 学习——与心理学家关于动物如何在这些条件下学习的假设非常相似。

13.15 延迟强化

影响定律要求对关系产生一种落后的影响，一些早期的法律批评家无法想象现在会如何影响过去的事物。当一个行为与随后的奖励或惩罚之间有相当大的延迟时，学习甚至可能发生，这一事实进一步加剧了这种担忧。类似地，在经典条件作用下，当我们的开始与 CS 相抵消时，学习也会发生。我们称之为延迟强化问题，这与明斯基 (1961) 所说的“学习系统的信贷分配问题”有关:how do you

将成功的荣誉分配到许多可能涉及到的决策中? 本书介绍的强化学习算法包括两个基本的机制来解决这个问题。第一个是使用资格跟踪，第二个是使用 TD 方法来学习价值函数，这些函数提供几乎即时的行为评估 (在工具条件作用实验中) 或者提供即时预测目标 (在经典条件作用实验中)。这两种方法都对应于动物学习理论中提出的类似机制。Pavlov(1927) 指出，每一种刺激都必须在神经系统中留下一个在刺激结束后持续一段时间的痕迹，他提出，当 CS 抵消和美国发作之间存在时间间隔时，刺激痕迹使学习成为可能。直到今天，在这些条件下的条件作用被称为跟踪条件作用 (第 344 页)。假设当美国到达时 CS 的踪迹仍然存在，那么通过跟踪和美国同时存在，就可以进行学习。我们在第十五章讨论了神经系统中微量机制的一些建议。刺激痕迹也被提出作为一种手段，以弥合行动之间的时间间隔和结果奖励或惩罚在工具性条件作用。例如，在赫尔影响深远的学习理论中，“磨牙刺激痕迹”解释了他所说的动物的目标梯度，描述了工具条件反应的最大强度是如何随着强化延迟的增加而降低的 (赫尔，1932,1943)。赫尔假设，动物的行为留下了内部刺激，其痕迹随着时间的作用而呈指数衰减。通过观察当时的动物学习数据，他假设在 30 到 40 秒后，这些痕迹会有效地达到零。本书中描述的算法中使用的资格跟踪类似于赫尔的跟踪: 它们是过去状态访问或过去状态操作对的衰减跟踪。Klopf(1972) 在他的神经元理论中引入了资格追踪，在这一理论中，它们是突触间过去活动的时间延伸，神经元之间的连接。Klopf 的轨迹比我们的算法使用的指数衰减轨迹更复杂，当我们在第 15.9 节讨论他的理论时，我们会进一步讨论这个问题。赫尔 (Hull)(1943) 提出，由于条件强化与磨牙刺激痕迹的共同作用，使目标梯度从目标向后传递，从而产生了更长的梯度。动物实验表明，如果条件有利于条件强化在延迟期的发展，那么学习就不会像在阻碍二次强化的条件下那样随着延迟的增加而减少。如果在延迟间隔期间有经常发生的刺激物，则条件强化更受欢迎。然后就好像奖励并没有因为有更直接的条件强化而被延迟。赫尔因此设想有一个主要的梯度，基于由刺激痕迹介导的初级强化的延迟，这是逐步修改和延长，由条件强化。在这本书中提出的算法使用资格跟踪和价值函数使学习延迟强化对应于赫尔的假设如何

动物能够在这些条件下学习。在第 13.5 节、第 15.7 节和第 15.8 节中讨论的演员-批评家架构最清楚地说明了这种对应关系。批评家使用 TD 算法来学习与系统当前行为相关联的值函数，即预测当前策略的返回值。演员根据评论家的预测更新当前的政策，或者更准确地说，基于评论家的预测的变化。评论家产生的 TD 错误作为一个条件强化信号，为参与者提供即时的绩效评估，即使主要奖励信号本身被严重延迟。评估动作价值函数的算法，如 Q-learning 和 Sarsa，同样使用 TD 学习原理，通过条件强化的方法使学习延迟强化。我们在第 15 章讨论的 TD 学习和多巴胺产生神经元的活动之间的紧密联系为强化学习算法和赫尔学习理论的这一方面提供了额外的支持。

13.16 认知地图

基于模型的强化学习算法使用环境模型，这些模型与心理学家所说的认知地图有共同的元素。回忆的规划和学习我们所讨论的在第 8 章, 通过一个环境模型我们意味着任何一个代理可以用来预测其环境将如何应对其行为状态转换和奖励, 我们所指的规划过程, 计算一个政策从这样一个模型。环境模型由两部分组成: 状态转换部分编码关于行为对状态变化的影响的知识，奖励模型部分编码关于每个状态或每个状态-动作对期望的奖励信号的知识。基于模型的算法通过使用一个模型来选择行为，根据未来的状态和预期的奖赏信号来预测可能的行为过程的结果。最简单的计划是比较“想象”的一系列决策集合的预测结果。关于动物是否使用环境模型的问题，如果使用环境模型，模型是什么样子的以及它们是如何学习的，在动物学习研究的历史中发挥了重要的作用。一些研究人员挑战了当时流行的学习和行为的刺激反应 (S-R) 观点，这与最简单的无模型的学习策略相对应，通过展示潜在的学习。在最早的潜在性学习实验中，两组大鼠在迷宫中奔跑。实验组在实验的第一阶段没有奖励，但是在第二阶段开始的时候，食物突然被引入了迷宫的目标盒中。对于对照组，在两个阶段，食物都在目标框中。问题是，在没有食物奖励的情况下，实验组的老鼠在第一阶段是否会学到任何东西。虽然实验大鼠在第一次实验中似乎没有学到很多东西，但当他们发现第二阶段引入的食物后，他们很快就发现了对照组的老鼠。结论是“在非奖励期，[实验组] 的大鼠发展出一种对迷宫的潜在性学习，一旦奖励被引入，他们就能利用这些潜在性学习”(Blodgett, 1929)。

潜伏学习是最密切相关的心理学家爱德华 • 杜尔曼解释这个结果, 和其他类似, 表明动物可以学习环境的“认知地图”没有奖励或处罚„ 他们可以使用地图后, 他们有动机去实现一个目标 (杜尔曼,1948)。一份认知地图也可以让老鼠计划一条通往目标的路线，而这条路线与老鼠在最初的探索中所使用的路线不同。对这些结果的解释导致了长期存在的争议，这是行为主义者/认知心理学二分法的核心。现代意义上，认知地图并不局限于空间布局的模型，而是更普遍的环境模型，或者是动物“任务空间”的模型 (如 Wilson, Takahashi, Schoenbaum, and Niv, 2014)。潜在学习实验的认知地图解释类似于动物使用基于模型的算法，环境模型即使没有明确的奖励或惩罚也可以学习。当动物被奖励或惩罚的表象所激励时，模型被用于计划。托尔曼对动物如何学习认知地图的描述是，当动物探索环境时，它们通过经历一连串的刺激来学习刺激-刺激或 S-S 联系。在心理学中，这被称为期望理论: 考虑到 S-S 关联，刺激的出现会产生下一个刺激的预期。这很像控制工程师所说的系统识别，在系统识别中，一个不知道动态的系统模型是从标记的训练例子中学到的。在最简单的离散时间版本中，训练例子是 S-S ? 对，S 是状态，S 是什么?，其后的状态是标签。当观察到 S 时，模型产生了 S 的“期望”。将观察到的下一个。对计划更有用的模型也包括操作，因此示例看起来像 SA-S ?, 年代在哪里? 当行为 A 在状态 s 中执行时，它是被期望的。在本例中，例子是形式 S - R 或 SA - R，其中 R 是与 S 或 SA 对相关联的奖励信号。这些都是监督学习的形式，通过这些形式，agent 可以获得认知地图，无论它在探索它的环境时是否接收到任何非零的奖励信号。

13.17 习惯性和目标导向的行为

无模型强化学习算法和基于模型的强化学习算法之间的区别与心理学家对习得行为模式的习惯控制和目标导向控制之间的区别是一致的。习惯是由适当的刺激触发的行为模式，然后或多或少地自动执行。根据心理学家使用这个短语的方式，目标导向行为是有目的性的，因为目标导向行为是由目标的价值以及行为与后果之间的关系所决定的。习惯有时被认为是由先行刺激控制的，而目标导向的行为则被认为是由结果控制的 (Dickinson, 1980, 1985)。目标导向控制的优势在于，当环境改变动物对其行为的反应方式时，它能迅速改变动物的行为。当习惯行为对来自于习惯环境的输入做出快速反应时，它却无法快速地适应环境的变化。目标导向的发展

行为控制可能是动物智力进化的一个重大进步。图 14.5 展示了假设任务中无模型和基于

模型的决策策略之间的区别，在这个假设任务中，老鼠必须在一个有不同目标框的迷宫中行走，每一个目标框都提供相应的大小奖励 (图 14.5 顶部)。从 S1 开始，老鼠必须首先选择左 (L) 或右 (R)，然后在 S2 或 S3 再次选择 L 或 R，以达到目标框之一。目标框是每一集老鼠情景任务的最终状态。无模型策略 (图 14.5 左下) 依赖于状态操作对的存储值。这些操作值是对每个 (非终端) 状态下的每个操作的最高回报的估计。他们是从开始到结束运行迷宫的许多试验中获得的。当行为值已足够好地估计最优回报时，大鼠只需在每个状态中选择动作值最大的动作，以实现最优

模范自由基于模型的

图 14.5: 基于模型和无模型的策略，以解决假设的顺序动作选择问题。上图: 一只老鼠用不同的目标盒子导航迷宫，每一个盒子都有一个奖励。左下: 无模型策略依赖于在许多学习试验中获得的所有状态-动作对的存储操作值。来做决定 rat 只需要在每个状态中选择动作值最大的动作。较低的正确示例: 在基于模型的策略中，老鼠学习了一个由知识组成的环境模型状态-行动-下一个状态的转换和一个由奖励知识组成的奖励模型与每个不同的目标框相关联。大鼠可以通过使用模型来模拟一系列的动作选择来决定在每个状态下转向的方式，从而找到一条产生最高的路径回报。改编自认知科学的趋势，第 10 卷，第 8 期，y[和合]，D. Joel 和 P。《动机的规范性视角》，2006 年第 376 页，获得爱思唯尔的许可。

决策。在这种情况下，当动作值估计值变得足够精确时，大鼠从 S1 和 R 中选择 L，以获得 4 的最大返回值。不同的无模型策略可能仅仅依赖于缓存的策略而不是动作值，从 S1 到 L、从 S2 到 r 进行直接链接。不需要参考状态转换模型，目标框的特性和它们所提供的回报之间不需要任何联系。图 14.5(右下) 演示了一个基于模型的策略。它使用由状态转换模型和奖励模型组成的环境模型。状态转换模型显示为一个决策树，而奖励模型将目标框的不同特征与每个目标框中的奖励联系起来。(与状态 S1、S2 和 S3 相关的奖励也是奖励模型的一部分，但这里它们是零，没有显示出来。) 基于模型的代理可以通过使用模型来模拟操作选择序列来确定在每个状态下转向的方式，从而找到一条产生最高回报的路径。在这种情况下，返回是在路径的最后得到的结果。在这里，有一个足够精确的模型，大鼠会选择 L 和 R 来获得 4 的奖励。比较模拟路径的预测收益是一种简单的规划形式，可以通过各种方式进行，如第 8 章所述。当无模型代理的环境改变了它对代理行为的反应方式时，代理必须在更改的环境中获得新的经验，在此期间它可以更新策略和/或值函数。模范自由策略如图 14.5(左下), 例如, 如果一个目标框以某种方式转向提供不同的奖励, 河鼠必须遍历迷宫, 可能很多时候, 体验新的奖励到达这一目标框, 同时更新其政策或其行为价值功能 (或两者) 在此基础上体验。关键的一点是，对于一个无模型的代理来说，要更改其策略为一个状态指定的动作，或者要更改与一个状态关联的动作值，它必须移动到那个状态，可能多次从那个状态开始操作，并体验其动作的后果。一个基于模型的代理可以适应其环境的变化，而不需要这种“个人体验”与受变化影响的状态和行为。模型的变更 (通过计划) 自动地改变它的策略。计划可以确定环境变化的后果，这些变化在代理自己的经验中从来没有联系在一起。例如，再一次提到图 14.5 的迷宫任务，想象一下，有一个先前学习过的转换和奖励模型的老鼠被直接放在 S2 的右边的目标框中，以发现现在可用的奖励值是 1 而不是 4。老鼠的奖励模式将会改变，即使在迷宫中寻找目标框所需要的行动选择没有被涉及。规划过程将为迷宫的运行带来新的奖励知识，无需在迷宫中增加额外的经验; 在本例中，将策略更改为右转，即 S1 和 S3，以获得 3 的返回值。这种逻辑正是动物的结果贬值实验的基础。这些实验的结果提供了动物是否已经学会了一种习惯，或者它的行为是否处于目标导向的控制之下。结果贬值实验就像后学实验，奖励会从一个阶段变化到下一个阶段。后

在学习的最初奖励阶段，结果的奖励值会发生变化，包括变成零，甚至变成负值。这一类型的早期重要实验是由 Adams 和 Dickinson(1981) 进行的。他们通过仪器调节训练大鼠，直到大鼠在训练室内用力按压蔗糖颗粒的杠杆。然后，老鼠被放置在同一个房间里，用操纵杆收回并允许非偶然的食物，这意味着球团可以独立于他们的行动提供给他们。在 15 分钟的自由接触后，一组老鼠被注射了令人作呕的氯化锂。这一实验重复了三次，在最后一次实验中，注射的老鼠都没有消耗任何非偶发颗粒，这表明小球的奖励价值已经降低——小球被贬值了。

在一天后进行的下一阶段中，老鼠再次被放置在实验室内，进行了一段灭种训练，这意味着反应杆已恢复正常，但与颗粒分发器断开连接，这样按压它就不会释放出颗粒。问题是，具有小球奖励值降低的大鼠是否会比没有小球奖励值的大鼠的杠杆作用更小，即使没有由于杠杆作用而导致的奖励值降低。结果显示，从物种灭绝试验开始，注射的老鼠的反应速度明显低于未注射的老鼠。亚当斯和狄金森得出结论，注射的大鼠通过认知图谱将杠杆按压与小球相联系，而小球则与恶心相关联。因此，在物种灭绝试验中，老鼠“知道”按下杠杆的后果是它们不想要的，因此它们从一开始就减少了杠杆的压力。重要的一点是，他们减少了杠杆的压力，而没有经历过直接的压力，然后是生病: 当他们生病的时候没有杠杆。他们似乎能够将行为选择的结果 (按下杠杆就会得到一个小球) 与结果的回报价值 (小球是要避免的) 结合起来，因此可以相应地改变他们的行为。并不是每个心理学家都同意这种“认知”的实验解释，它不是解释这些结果的唯一可能的方法，但是基于模型的规划解释被广泛接受。没有什么可以阻止代理同时使用无模型和基于模型的算法，并且有充分的理由同时使用这两种算法。我们从自己的经验中知道，只要有足够的重复，目标导向的行为就会变成习惯性的行为。实验表明，这种情况也发生在老鼠身上。Adams(1982) 进行了一项实验，看看扩展训练是否能将目标导向的行为转化为习惯性行为。他通过比较结果贬值对经历不同程度训练的老鼠的影响来做这个实验。如果与接受较少训练的大鼠相比，长期训练使大鼠对货币贬值更不敏感，这将是长期训练使行为更习惯性的证据。Adams 的实验紧跟着 Adams 和 Dickinson(1981) 的实验。稍微简化一下，一组老鼠被训练到 100 个奖励的杠杆，另一组老鼠被训练到 500 个奖励的杠杆。经过这次训练，小白鼠的球团的奖励值降低 (使用氯化锂注射)。

在两组。然后两组大鼠都接受了灭绝训练。亚当斯的问题是，贬值会不会对训练过度的老鼠产生比没有训练过度的老鼠更少的杠杆作用，这将是延长训练减少对结果贬值的敏感性的证据。结果证明，贬值大大降低了未受过过度训练的大鼠的压杠杆率。相反，对于训练过度的老鼠来说，贬值对它们的杠杆作用不大; 事实上，如果说有什么不同的话，那就是它更有活力。(完整的实验包括对照组，显示不同程度的训练本身并没有显著影响学习后的杠杆率。) 这一结果表明，尽管未受过过度训练的大鼠对自己行为的结果非常敏感，但训练过度的大鼠却养成了一种施加压力的习惯。从计算的角度来看这个和其他类似的结果，就可以理解为什么人们可能会期望动物在某些情况下习惯性地行动，在其他情况下以目标导向的方式行动，以及为什么它们在继续学习时从一种控制模式转向另一种控制模式。毫无疑问，动物使用的算法与我们在本书中展示的算法并不完全匹配，但人们可以通过考虑各种强化学习算法所暗示的权衡来深入了解动物的行为。计算神经学家 Daw、Niv 和 Dayan(2005) 提出了一个想法，即动物使用无模型和基于模型的过程。每个过程都提出一个动作，而选择执行的动作是由被认为是两个过程中更值得信赖的过程提出的，这是由贯穿整个学习过程的信心度量所决定的。早期学习基于模型的系统的计划过程更加值得信赖，因为它将短期的预测联系在一起，而这些短期的预测比没有模型的过程的长期预测更准确。但是，随着经验的不断积累，无模型的过程变得更加值得信赖，因为规划很容易出错，因为模型的不准确性和使规划可行所必需的捷径，例如各种形式的“树修剪”: 删除无前景的搜索树分支。根据这个观点，随着经验的积累，人们会期望从目标导向的行为转向习惯行为。关于动物如何在目标导向和习惯性控制之间进行仲裁，人们提出了其他的观点，行为和神经科学研究都在继续研究这个问题和相关的问题。无模型和基于模型的算法之间的区别被证明对这项研究是有用的。我们可以在抽象设置中检查这些类型的算法的计算含义，以揭示每种类型的基本优点和局限性。这既可以提出建议，也可以使问题变得更加尖锐，这些问题可以指导实验的设计，以增加心理学家对习惯性和目标导向行为控制的理解。

13.18 总结

本章的目的是探讨强化学习与动物学习心理学实验研究之间的对应关系。我们在一开始就强调，本书所描述的强化学习并不是有意的为动物行为的细节建模。它是一个抽象的计算框架，

从人工智能和工程的角度探索理想化的情况。但是，许多基本的强化学习算法都是受到心理学理论的启发，在某些情况下，这些算法促进了新的动物学习模式的发展。这一章描述了这些通信中最明显的一个。

预测算法和控制算法之间的强化学习的区别与动物学习理论在经典条件作用和工具条件作用之间的区别类似。工具性和经典条件作用实验的关键区别在于，前者的强化刺激取决于动物的行为，而后者则不然。通过 TD 算法学习预测与经典条件作用相对应，我们将经典条件作用的 TD 模型描述为一个例子，强化学习原理解释了动物学习行为的一些细节。这个模型概括了有影响的 Rescorla-Wagner 模型，它包含了在个体试验中事件影响学习的时间维度，并且它提供了二阶条件作用的解释，在二阶条件作用中，强化刺激的预测因子会自我强化。这也是对大脑中多巴胺神经元活性有影响的观点的基础，这是我们在第 15 章中所讨论的。试错学习是强化学习控制的基础。我们在这里和第 1 章 (第 15 页) 中讨论了索恩代克用猫和其他动物做实验的一些细节。指出在强化学习中，探索并不局限于“盲目摸索”; 只要有一些探索，试验就可以通过使用先天和以前学到的知识的复杂方法产生。我们讨论了训练方法 b • f • 斯金纳 (B. F. Skinner) 所称的“塑形”，在这个过程中，奖励或突发事件被逐步改变，以训练一种动物连续地接近所期望的行为。塑形不仅是动物训练中不可缺少的，也是训练强化学习因子的有效工具。这也与动物的动机状态有关，它会影响动物接近或避免什么，以及什么事件会对动物产生奖励或惩罚。本书介绍的强化学习算法包括两种基本机制来解决延迟强化问题: 通过 TD 算法学习的资格跟踪和价值函数。这两种机制在动物学习理论中都有先例。资格追溯类似于早期理论的刺激痕迹，价值函数对应于次要强化作用，提供几乎即时的评价反馈。下一章的内容是在强化学习的环境模型和心理学家所说的认知地图之间。20 世纪中期进行的实验旨在展示动物学习认知地图的能力，将其作为国家行动协会的替代品，或作为补充，然后利用它们指导行为，特别是当环境发生意外变化时。强化学习中的环境模型就像认知地图，它们可以通过监督学习方法学习而不依赖奖励信号，然后它们就可以

后来用于计划行为。强化学习在无模型和基于模型的算法之间的区别与习惯性和目标导向行为之间的心理学区别。无模型算法通过访问存储在策略或动作值函数中的信息来做出决策，而基于模型的方法则通过使用代理环境的模型提前规划来选择操作。结果-贬值实验提供了动物行为是习惯性还是被目标控制的信息。强化学习理论有助于澄清对这些问题的思考。动物学习清楚地告知强化学习，但作为一种机器学习，强化学习的目的是设计和理解有效的学习算法，而不是复制或解释动物行为的细节。我们专注于动物学习相关方面的明确方法解决预测和控制问题的方法, 强调了丰硕的强化学习和心理学之间的双向流动的思想没有冒险深入的许多行为细节和争议占据了动物学习研究人员的注意。强化学习理论和算法的未来发展很可能会利用与动物学习的许多其他特性的联系，因为这些特性的计算效用会得到更好的理解。我们期望在强化学习和心理学之间的思想交流将继续为这两个学科带来成果。强化学习与心理学和其他行为科学领域之间的许多联系超出了本章的范围。我们在很大程度上讨论了与决策心理学的联系，决策心理学关注的是在学习之后如何选择行动或如何做出决策。我们也不讨论生态学家和行为生态学家所研究的行为的生态和进化方面的联系: 动物如何相互联系和它们的物理环境，以及它们的行为如何有助于进化适应。优化、MDPs 和动态编程在这些领域中占据着重要地位，我们强调与动态环境的代理交互，这与研究复杂的“生态论”中的代理行为有关。多主体强化学习，在这本书中被省略了，与行为的社会方面有联系。尽管缺乏治疗，强化学习绝不应该被解释为忽视进化的观点。强化学习并不意味着学习和行为有一张白板。事实上，工程应用的经验强调了构建强化学习系统知识的重要性，这与进化为动物提供的知识类似。

13.19 书目的和历史的言论

Ludvig, Bellemare 和 Pearson(2011) 和 Shah(2012) 回顾了心理学和神经科学语境下的强化学习。这些出版物是本章和下一章关于强化学习和神经科学的有用的伙伴。

14.1 Dayan, Niv, Seymour 和 Daw(2006) 专注于 clas-之间的交互物理条件作用和工具条件作用，特别是当分类条件作用和工具反应发生冲突时。他们提出了一个 Q-learning 框架来建模这种交互的各个方面。Modayil 和 Sutton(2014) 利用一个移动机器人展示了将固定响应与在线预测学习相结合的控制方法的有效性。他们把这种方法称为“pavlovian 控制”，强调它与通常的强化学习控制方法不同，这种方法是建立在正确执行固定反应的基础上，而不是基于奖励最大化。罗斯 (1933) 的机电机器，特别是沃尔特的海龟 (沃尔特，1951 年) 的学习版本是巴甫洛夫控制的早期插图。

14.2.1 Kamin(1968) 首先报道了阻塞，现在通常被称为 Kamin 阻塞，在经典条件作用。Moore 和 Schmajuk(2008) 对阻塞现象、它所刺激的研究及其对动物学习理论的持久影响进行了很好的总结。Gibbs, Cool, Land, Kehoe 和 Gormezano(1991) 描述了兔子的瞬态膜反应的二级条件反射，以及它与连续复合刺激作用的关系。芬奇 (Finch and Culler)(1934) 报告说，“当动物的动机通过不同的指令得到维持时”，狗的前肢退缩得到了五阶条件反射。

14.2.2 Rescorla-Wagner 模型的想法是，学习发生在动物身上《惊奇》改编自 Kamin(1969)。除了 Rescorla 和 Wagner 之外的古典条件作用模型包括 Klopf(1988)、Grossberg(1975)、Mackintosh(1975)、Moore and Stickney(1980)、Pearce and Hall(1980)、Courville、Daw 和 Touretzky(2006)。Schmajuk(2008) 回顾经典条件作用模型。Wagner(2008) 对 Rescorla-Wagner 模型和类似的学习基本理论提供了现代心理学视角。

14.2.3 经典条件作用的 TD 模型的早期版本出现在萨顿和 Barto (1981a)，也包括了早期模型的预测，即时间主导压倒了阻塞，后来 Kehoe、Schreurs 和 Graham(1987) 等人证明了这一点，即在兔裂膜制备过程中发生。萨顿和巴托 (1981a) 对 Rescorla - Wagner 模型和最小均方 (LMS) 或 Widrow-Hoff，学习规则 (Widrow 和 Hoff, 1960) 之间的近同一性进行了最早的认识。早期的模型在 Sutton 的 TD 算法 (Sutton, 1984, 1988) 之后进行了修订，并在 Sutton 和 Barto(1987) 中首次提出 TD 模型，在 Sutton 和 Barto(1990) 中更全面，这部分主要基于此部分。额外的探索

TD 模型及其可能的神经实现由 Moore 和他的同事 (Moore, Desmond, bertier, Blazis, Sutton, and Barto, 1986; 摩尔和 Blazis,1989;Moore, Choi, and Brunzell, 1998; 摩尔，马克斯，卡斯塔尼亚，波勒万，2001)。Klopf(1988) 经典条件作用的驱动强化理论扩展了 TD 模型以解决额外的实验细节，如采集曲线的 s 型。在这些出版物中，TD 被认为是时间导数，而不是时间差异。

14.2.4 Ludvig、Sutton 和 Kehoe(2012) 对 TD 模型的性能进行了评价在先前未探索过的涉及古典条件作用的任务中，研究了各种刺激表现形式的影响，包括他们早先引入的微刺激表现形式 (Ludvig, Sutton, and Kehoe, 2008)。在 TD 模型的背景下，关于各种刺激表示的影响及其可能的神经实现对响应时间和地形的影响的早期研究是摩尔和他的同事在上面提到的。虽然不是在 TD 模型的背景下，但是像 Ludvig 等人 (2012) 的微刺激表征已经被 Grossberg 和 Schmajuk(1989)、Brown、Bullock 和 Grossberg(1999)、Buhusi 和 Schmajuk(1999) 以及 Machado(1997) 提出和研究。第 353-355 页的数字改编自萨顿和巴托 (1990)。

14.3 第 1.7 节包括关于试错学习历史的评论的法律效果。索恩代克的猫可能是根据直觉的特定环境而不是行动的顺序来探索的，而不是仅仅从一组本能的冲动中选择。Selfridge, Sutton, and Barto(1985) 展示了在杆平衡强化学习任务中塑造的有效性。在强化学习的其他例子塑造 Gullapalli 和 Barto(1992), 马哈和康奈尔大学 (1992), 马塔里奇 (1994), 民宿和 Colombette(1994),Saksida, 雷蒙德, 和 Touretzky(1997), 和 Randløv Alstrøm(1998)。Ng(2003)、Ng、Harada 和 Russell(1999) 在某种意义上使用了与 Skinner 不同的“塑造”一词，关注的是如何在不改变最优策略的情况下改变奖励信号的问题。Dickinson 和 Balleine(2002) 讨论了学习和动机之间相互作用的复杂性。Wise(2004) 概述了强化学习及其与动机的关系。Daw 和 Shohamy(2008) 将动机和学习联系到强化学习理论的各个方面。参见《麦克卢尔》、《杜》、《蒙太古》(2003)、《和合》、《乔尔》、《大安》(2006)、《兰格尔》、《卡默尔》、《蒙太古》(2008)、《大安》、《贝里奇》(2014)。McClure 等人 (2003)，Niv, Daw, Daw, Dayan (2006)，and Niv, Daw, Joel, Dayan(2007) 提出了与强化学习框架相关的行为活力理论。

14.4 斯宾塞，赫尔的学生和耶鲁大学的合作者，阐述了更高的角色。在解决延迟强化问题 (Spence, 1947) 的问题上加强了秩序。学习很长时间的延迟，如在厌恶的条件下延迟数小时，导致干涉理论取代衰减跟踪理论 (例如，Revusky 和 Garcia, 1970; 伯克和科斯塔,2014)。在延迟强化下学习的其他观点引用了意识和工作记忆的角色 (例如 Clark 和 Squire, 1998;Seo, Barraclough，和 Lee, 2007)。

14.5 Thistlethwaite(1951) 对潜在学习实验进行了广泛的回顾直到它出版的时候。Ljung(1998) 提供了模型学习的概述，或系统识别，工程技术。Gopnik, Glymour, Sobel, Schulz, Kushnir 和 Danks(2004) 提出了一个关于儿童如何学习模型的贝叶斯理论。

14.6 习惯行为和目标导向行为以及无模型行为之间的联系基于模型的强化学习最早由 Daw、Niv 和 Dayan(2005) 提出。用来解释习惯性和目标导向行为控制的假设迷宫任务是基于对 Niv, Joel，和 Dayan(2006) 的解释。Dolan 和 Dayan(2013) 回顾了与这个问题相关的四代实验研究，并讨论了如何在强化学习无模型/基于模型的区别的基础上继续前进。Dickinson(1980, 1985) 和 Dick-inson 和 Balleine(2002) 讨论了与这一区别有关的实验证据。Donahoe 和 Burgos(2000) 也认为，无模型过程可以解释收益-贬值实验的结果。Dayan 和 Berridge(2014) 认为经典条件反射涉及基于模型的过程。Rangel、Camerer 和 Montague(2008) 回顾了许多未解决的问题，包括习惯性、目标导向和 Pavlovian 控制模式。

评论心理学术语——传统意义上的钢筋的强化的行为模式 (通过增加其强度或频率) 由于动物接收刺激 (或体验刺激的遗漏) 在一个合适的时间与另一个刺激和反应之间的关系。强化产生的变化留在未来的行为中。在心理学中，强化有时是指在行为上产生持久变化的过程，无论这些变化是加强还是削弱了一种行为模式 (Mackintosh, 1983)。强化指的是强化之外的弱化，这与强化的日常意义，以及它在心理学中的传统用法不一致，但它是我们在这里采用的一个有用的扩展。在任何一种情况下，被认为是行为改变的原因的刺激被称为强化物。心理学家通常不像我们一样使用特定的短语强化学习。动物学习先驱者可能把强化和学习视为同义词，因此使用这两个词是多余的。我们在计算和工程研究中使用这个短语，主要受明斯基 (Minsky, 1961) 的影响。但这个短语最近在心理学和神经科学中越来越流行，很可能是因为强化学习算法和动物学习算法之间出现了强烈的相似之处——这一章和下一章中所描述的相似之处。根据通常的用法，奖励是动物接近和工作的对象或事件。动物因其“好”而受到奖励

为了使动物的行为更好而给予的行为。类似地，惩罚是动物通常避免的对象或事件，是“不良”行为的结果，通常是为了改变这种行为。主要奖励奖励是由于机器内置一个动物神经系统的进化, 以改善其生存和繁殖的机会, 例如, 奖励由营养食物的味道, 性接触、成功逃脱, 和许多其他的刺激和事件, 预测在动物繁殖成功率的祖先历史。如第 14.2.1 节所解释的，高阶奖励是通过预测初级奖励的刺激物提供的奖励，直接或间接地通过预测其他预测初级奖励的刺激物。如果奖励质量是直接预测主要奖励的结果，那么奖励是次要的。在这本书中，我们称 Rt 为“t 时刻的奖励信号”，有时仅仅是“t 时刻的奖励信号”，但我们不认为它是 agent 环境中的对象或事件。因为 Rt 是一个数字——不是物体或事件——它更像是神经科学中的奖赏信号，它是大脑内部的信号，就像神经元的活动一样，影响决策和学习。当动物感知到一个有吸引力的 (或令人厌恶的) 物体时，这个信号可能会被触发，但它也可能被一些在动物外部环境中不存在的东西触发，比如记忆、想法或幻觉。因为我们的 Rt 可以是正的，负的，或零的，所以我们最好把负的 Rt 叫做惩罚，Rt 等于零是一个中性信号，但为了简单起见，我们通常避免这些项。在强化学习中，生成所有 Rts 的过程定义了代理试图解决的问题。代理人的目标是在一段时间内尽可能地保持 Rt 的大小。Rt 是在这方面, 主要奖励一个动物, 如果我们认为动物面临的问题的问题在其生命周期中获得尽可能多的主要奖励 (, 因此, 通过进化的未来的“智慧”, 提高解决实际问题的机会, 这是其基因传递给后代)。然而，正如我们在第 15 章中所指出的，在动物的大脑中不太可能存在一个像 Rt 这样的“主人”奖励信号。不是所有的强化物都是奖励或惩罚。有时，强化不是动物接受刺激的结果，刺激是通过给行为贴上好或坏的标签来评估其行为。一种行为模式可以通过刺激得到强化，无论动物的行为如何。如第 14.1 节所述，强化物的提供是否取决于，或不取决于，前一行为是工具的，或操作的，条件作用实验和经典的，或巴甫洛夫的，条件作用实验的决定性区别。在这两种实验中，强

化都是有效的，但只有前者是对过去行为进行评估的反馈。(尽管它经常被指出, 即使在加强我们在经典条件反射实验中不是取决于主体的前行为, 强化价值可以通过这种行为, 影响的一个例子是, 一个封闭的眼睛使眼睛不那么厌恶空气膨胀。) 当我们在下一章讨论这些信号的神经关联时，奖励信号和强化信号之间的区别是一个关键点。就像奖励信号，对我们来说，任何特定时间的强化信号都是正数或负数，或零。强化信号是指导学习算法变化的主要因素

在代理的策略、价值评估或环境模型中生成。对我们最有意义的定义是，增强信号在任何时候都是一个数字，它乘以一个向量 (可能还有一些常数) 来确定某些学习算法中的参数更新。对于某些算法来说，奖励信号本身就是参数更新方程的关键乘法器。对于这些算法，增强信号与奖励信号相同。但对大多数的算法我们讨论在本书中, 强化信号包括条款除了奖励信号, 一个例子是 TD 错误 $\mathrm{t} = \mathrm{Rt} + 1 + \mathrm{V}(圣 + 1) - \mathrm{V}(\mathrm{St})$, 即强化信号道明州值学习 (和类似的 TD 错误行为价值学习)。在这种强化信号,$\mathrm{Rt} + 1$ 是主要的强化的贡献, 和颞预测值的差异, $\mathrm{V}(圣 + 1) - \mathrm{V}(\mathrm{St})$(或类似的行动时间差异值), 条件是钢筋的贡献。因此, 每当 $\mathrm{V}(圣 + 1) - \mathrm{V}(\mathrm{St}) = 0$, t 信号“纯”主要强化; 当 $\mathrm{Rt}+1 = 0$ 时，它表示“纯”条件强化，但它通常表示这些的混合。注意我们 6.1 节中提到的, 这个 t 才可用时间 $\mathrm{t} + 1$。因此我们认为 t 强化信号在时间 $\mathrm{t} + 1$, 这是合适的, 因为它增强了预测和/或行为使 t 前一步。一个可能的混淆来源是著名心理学家 b • f • 斯金纳及其追随者使用的术语。对于斯金纳来说，当动物行为的结果增加了这种行为的频率时，就会产生积极的强化作用; 当行为的后果降低了行为的频率时，惩罚就会发生。当行为导致厌恶刺激 (即动物不喜欢的刺激) 被移除时，负面强化就会发生，从而增加这种行为的频率。另一方面，当行为导致食欲刺激 (即动物喜欢的刺激) 消失时，就会出现消极惩罚，从而降低这种行为的频率。我们发现没有必要对这些区别进行严格的区分，因为我们的方法比这更抽象，奖励和强化信号都允许同时具有积极和消极的价值。(但特别要注意，当我们的强化信号为负时，它与斯金纳的负强化并不相同。) 另一方面，人们也经常指出，仅仅依靠一个数字作为奖励或惩罚信号，这与动物的食欲和厌恶系统具有不同的性质和涉及不同的大脑机制这一事实是不一致的。这指向了一个方向，在这个方向上，未来的强化学习框架可能会被开发出来，以利用不同的欲望和厌恶系统的计算优势，但现在我们正在传递这些可能性。另一个术语上的差异是我们如何使用行动这个词。对于许多认知科学家来说，一种行为是有目的的，因为它是动物对问题行为和行为后果之间关系的认知的结果。行动是目标导向的，是决定的结果，而不是由刺激引发的反应; 反射或习惯的结果。我们使用“行动”这个词，而不区分其他人所谓的“行动”、“决定”和“反应”。这些是重要的区别，但对我们来说，它们是由不同的区别所包围的

无模型和基于模型的强化学习算法，我们在上面的 14.6 节中讨论了它们与习惯行为和目标导向行为的关系。Dickinson(1985) 讨论了反应和行动之间的区别。这本书中经常用到的一个词是“控制”。我们所说的控制与动物学习心理学家所说的完全不同。所谓控制，我们指的是代理影响其环境，从而导致代理喜欢的状态或事件: 代理对其环境施加控制。这是控制工程师使用的控制感。另一方面，在心理学中，控制通常意味着动物的行为受到动物受到的刺激 (刺激控制) 或它所经历的强化计划的控制。在这里，环境控制代理。在这种意义上的控制是行为矫正治疗的基础。当然，当代理与环境交互时，这两个控制方向都起作用，但是我们的关注点是代理作为控制器; 不是环境作为控制器。与我们的观点相对应，也许更有启发性的是，代理实际上控制了从其环境接收的输入 (Powers, 1973)。这不是心理学家所说的刺激控制。有时，强化学习被理解为仅仅从奖励 (和惩罚) 中直接学习策略，而不涉及价值函数或环境模型。这就是心理学家所说的刺激-反应或 S-R 学习。但对我们来说，和今天的大多数心理学家一样，强化学习远不止于此，除了 S-R 学习之外，还包括价值函数、环境模型、计划和其他通常被认为属于心理功能认知方面的方法。

14. 第 15 章神经科学

神经科学是对神经系统的多学科研究。它们如何调节身体机能；控制行为；随着时间的推移，由于发展、学习和衰老而发生的变化；细胞和分子机制如何使这些功能成为可能。强化学习最令人兴奋的一个方面是来自神经科学的越来越多的证据表明，人类和许多其他动物的神经系统实现了与强化学习算法相对应的算法。本章的主要目的是解释这些相似之处，以及它们对动物奖励相关学习的神经基础的建议。强化学习和神经科学之间最显著的联系涉及到多巴胺，这是一种化学物质，与哺乳动物大脑中的奖赏过程密切相关。多巴胺似乎将时间差异 *(TD)* 错误传递到学习和决策发生的大脑结构中。这种平行关系可以用多巴胺神经元活动的奖赏预测误差假说来表达，这一假说是由计算强化学习的收敛和神经科学实验的结果所导致的。在这一章中，我们讨论了这个假设，导致它的神经科学发现，以及为什么它是对理解大脑奖励系统的重要贡献。我们还讨论了强化学习和神经科学之间的相似之处，它们与多巴胺/ *td* -错误的平行度相比，并没有那么显著，但它提供了一些有用的概念性工具，可以用来思考动物身上基于奖励的学习。强化学习的其他元素有可能影响神经系统的研究，但它们与神经科学的联系仍相对不发达。我们讨论了一些我们认为随着时间的推移会变得越来越重要的进化关系。正如我们在这本书的第一章 *(第 1.7 节)* 的历史部分中所概述的，强化学习的许多方面都受到神经科学的影响。本章的第二个目标是让读者了解大脑功能的概念，这些概念对强化学习的方法有帮助。根据大脑功能理论，强化学习的一些要素更容易理解。这一点对资格追踪的观点尤其正确，它是强化学习的基本机制之一，起源于神经突触的一种推测性质，神经细胞——神经细胞——通过这种结构相互交流。

在这一章中，我们没有深入研究动物基于奖励的学习的神经系统的巨大复杂性：这一章太短，我们不是神经科学家。我们并没有试图去描述——甚至是说出——许多被认为与这些过程有关的大脑结构和通路，或者任何分子机制。我们也不公正的假设和模型是替代那些与强化学习非常一致的。在这一领域的专家之间存在着不同的观点，这并不奇怪。我们只能对这个引人入胜、不断发展的故事略知一二。尽管如此，我们希望这一章能让你们相信，一个非常有效的途径已经出现，它将强化学习及其理论基础与动物基于奖励的学习的神经科学联系起来。许多优秀的出版物涉及强化学习和神经科学之间的联系，其中一些我们在本章最后一节中引用。我们的治疗方法与大多数的不同，因为我们假设对强化学习的熟悉程度是在本书的前几章中提出的，但我们并没有假设神经科学的知识。我们先简单介绍一下神经科学的概念，以便对接下来的内容有一个基本的了解。

14.1 神经科学基础知识

关于神经系统的一些基本信息有助于我们在本章中介绍的内容。我们后面提到的术语是斜体的。如果你已经有了神经科学的基础知识，跳过这一部分就不是问题。神经元是神经系统的主要组成部分，是利用电信号和化学信号处理和传输信息的细胞。它们有很多种形式，但是神

经元通常有一个细胞体、树突和一个轴突。树突是一种结构，它可以从细胞的身体分支接收来自其他神经元的输入 (也可以在感觉神经元的情况下接收外部信号)。神经元的轴突是将神经元的输出传输到其他神经元 (或肌肉或腺体) 的纤维。一个神经元的输出包括一系列的电脉冲称为动作电位沿轴突运动。动作电位也称为脉冲，神经元在产生脉冲时就会发出脉冲。在神经网络模型中，通常使用实数来表示神经元的放电频率，即每一单位时间的平均峰值次数。神经元的轴突可以广泛地分支，使神经元的动作电位达到许多目标。神经元轴突的分支结构称为神经元轴突柄。因为动作电位的传导是一个活跃的过程，就像引信的燃烧一样，当动作电位到达轴向分支点时，它会“点亮”所有传出分支上的动作电位 (尽管传播到分支有时会失败)。因此，具有较大轴突杆的神经元的活动可以影响许多目标位点。

突触通常是位于轴突分支末端的一种结构，它调节一个神经元与另一个神经元之间的通信。突触将信息从突触前神经元的轴突传输到突触后神经元的树突或细胞体。除了少数例外，突触在突触前神经元的动作电位到达时释放出一种化学神经递质。(例外情况是神经元之间直接的电耦合，但这里我们不关心这些。) 突触的神经递质分子从突触前释放一边扩散在突触间隙非常小的空间之间的突触前和突触后神经元, 结束, 然后结合受体表面的突触后神经元的兴奋或抑制其 spike-generating 活动, 或以其他方式调整其行为。一种特殊的神经递质可能与几种不同类型的受体结合，每一种受体对突触后神经元产生不同的影响。例如，至少有五种不同的受体类型的神经递质多巴胺可以影响突触后神经元。许多不同的化学物质被确认为动物神经系统中的神经递质。神经元的背景活动活动的水平, 通常其燃烧速度, 当神经元似乎并不由突触输入实验者感兴趣的相关任务, 例如, 当神经元的活动不与刺激传递到一个主题作为一个实验的一部分。由于来自更广泛的网络的输入，或者由于神经元或其突触内部的噪音，背景活动可能是不规则的。有时，背景活动是神经元内在动力过程的结果。一个神经元的阶段性活动，与它的背景活动相反，是由突触输入引起的脉冲活动组成的。缓慢变化的活动，经常以渐进的方式变化，无论是作为背景活动还是不作为背景活动，都被称为神经元的滋补活动。突触释放的神经递质影响突触后神经元的强度或效力是突触的效力。神经系统可以改变通过经验的一种途径是通过改变突触的功效的结果组合突触前和突触后神经元的活动, 有时存在神经调质, 这是一种神经递质有影响以外, 或除了直接快速激发或抑制。大脑中有几个不同的神经调节系统，这些系统由神经元簇组成，这些神经元的轴突弧线分枝很广，每个系统都使用不同的神经递质。神经调节可以改变神经回路的功能，调节动机、觉醒、注意力、记忆、情绪、情绪、睡眠和体温。重要的是，神经调节系统可以分布一些像标量信号 (如增强信号) 这样的信号，以改变广泛分布的对学习至关重要的突触的操作。突触功能的改变被称为突触可塑性。它是负责学习的主要机制之一。通过学习算法调整的参数或权重对应于突触效率。正如我们在下面详细介绍的，通过神经调节器多巴胺调节突触可塑性是大脑如何实现学习算法的一个合理机制，就像本书中描述的那样。

14.2 奖励信号、强化信号、值和预测错误

神经科学和计算强化学习之间的联系开始于大脑信号和在强化学习理论和算法中扮演重要角色的信号之间的平行关系。在第三章中，我们说过任何学习目标导向行为的问题都可以归结为表示行为、状态和奖励的三个信号。然而，为了解释神经科学和强化学习之间的联系，我们必须不那么抽象，并考虑其他与大脑信号在某些方面相对应的强化学习信号。除了奖励信号外，这些信号还包括强化信号 (我们认为这与奖励信号不同)、值信号和传递预测错误的信号。当我们用函数来标记一个信号时，我们是在强化学习理论的背景下进行的，在强化学习理论中，信号对应于方程或算法中的一个项。另一方面，当我们提到大脑中的一个信号时，我们指的是一个生理事件，如动作电位的爆发或神经递质的分泌。用它的功能给神经信号贴上标签，例如把多巴胺神经元的阶段性活动称为强化信号，这意味着神经信号的表现与相应的理论信号相似。发现这些通信的证据涉及许多挑战。与奖励处理相关的神经活动几乎可以在大脑的每个部分中找到，而且很难明确地解释结果，因为不同奖赏相关信号的表征往往彼此高度相

关。实验需要仔细设计，以使一种与奖励相关的信号能够在一定程度上与其他信号区分开来，或者与奖励处理无关的大量其他信号区分开来。尽管存在这些困难，许多实验都是为了将强化学习理论和算法的各个方面与神经信号相协调，并建立了一些引人注目的联系。为了准备检查这些链接，在本节的其余部分，我们提醒读者，根据强化学习理论，各种奖励相关的信号意味着什么。在我们对前一章末尾的术语的评论中，我们说 Rt 就像动物大脑中的奖励信号，而不是动物环境中的一个物体或事件。在强化学习中，奖励信号 (以及代理环境) 定义了强化学习代理试图解决的问题。在这方面，Rt 就像动物大脑中的一个信号，将主要的奖励分配给整个大脑的各个部位。但是像 Rt 这样的单一的奖励信号不太可能存在于动物的大脑中。最好将 Rt 看作是一个抽象概念，它总结了大脑中许多系统产生的大量神经信号的总体影响，这些系统评估感觉和状态的奖励或惩罚性质。强化学习中的强化信号不同于奖励信号。增强信号的功能是指导学习算法在代理策略、价值估计或环境模型中所做的更改。TD 方法, 例如, 强化信号在时间 t TD 错误 t−1 = Rt + V(St)−V(圣 −1)。1 1 我们 6.1 节中提到的, t 在我们的符号定义是 Rt + 1 + V(圣 + 1)−V(St), 所以 t

某些算法的增强信号可能只是奖励信号，但对于大多数算法，我们认为增强信号是由其他信息调整的奖励信号，如 TD 误差的值估计。状态值或动作值的估计值，即 V 或 Q，指定从长期来看对代理是好是坏。他们预测的是一名经纪人在未来积累的总报酬。代理通过选择导致状态估计值最大的动作或选择行为估计值最大的动作来做出正确的决策。预测误差测量预期和实际信号或感觉之间的差异。奖励预测误差 (Reward prediction errors, RPEs) 专门测量预期和收到的奖励信号之间的差异，当奖励信号大于预期时为正值，反之为负值。像 (6.5) 这样的 TD 错误是一种特殊的 RPEs 类型，表明当前和早期的长期回报预期之间存在差异。当神经科学家提到 RPEs 时，他们通常 (虽然不是总是) 指 TD RPEs，我们简单地称之为 TD 错误贯穿本章。同样在本章中，TD 错误通常不依赖于动作，而不是像 Sarsa 和 Q-learning 这样的算法在学习动作值时使用的 TD 错误。这是因为与神经科学最著名的联系是在没有行动的 TD 错误上，但我们并不是说排除可能的类似的联系，包括与行动相关的 TD 错误。(TD 错误用于预测除了奖励以外的信号也很有用，但在这里我们并不关心这个问题。例如，Modayil, White 和 Sutton, 2014)。人们可以问很多关于神经科学数据和这些理论定义的信号之间的联系的问题。观察到的信号更像是奖励信号、价值信号、预测吗错误，强化信号，还是完全不同的东西? 如果是错误信号，是 RPE, TD 错误，还是像 Rescorla-Wagner 错误 (14.3) 这样简单的错误? 如果是 TD 错误，它是否依赖于像 Q-learning 或 Sarsa 的 TD 错误这样的行为? 如上所示，探索大脑来回答这样的问题是极其困难的。但实验证据表明，一种神经递质，特别是神经递质多巴胺，会向 RPEs 发送信号，进一步说，多巴胺产生的神经元的阶段活动实际上传递了 TD 错误 (关于阶段活动的定义，请参阅第 15.1 节)。这一证据导致了我们接下来描述的多巴胺神经元活动的奖励预测错误假说。

14.3 奖励预测误差假设

多巴胺神经元活动的奖赏预测误差假说提出，哺乳动物产生多巴胺的神经元的阶段性活动的功能之一，是在旧的和新的对预期未来奖赏的估计之间，将错误传递给整个大脑的目标区域。这个假设 (虽然不是用这些确切的词) 是由 Montague、Dayan 和 Sejnowski(1996) 首次明确提出的，他们展示了增强学习中的 TD 错误概念如何解释了阶段性的许多特征

直到 t+ 1 时才可用。TD 的错误可以在 t 是 t−1 = Rt + V(St)−V(圣 −1)。因为我们认为时间步是非常小的，甚至是无穷小的，时间间隔，我们不应该把这一步的时间间隔看得太重要。

哺乳动物多巴胺神经元的活性。导致这一假设的实验是在 20 世纪 80 年代和 90 年代早期在神经学家 Wolfram Schultz 的实验室中进行的。第 15.4 节描述了这些有影响的实验，第 15.6 节解释了这些实验的结果如何与 TD 错误一致，本章末尾的书目和历史注释部分包含了关于这个有影响的假设的发展的文献指南。Montague 等 (1996) 将经典条件作用的 TD 模型

的 TD 误差与经典条件作用下产生多巴胺的神经元的相活动进行了比较。回想一下 14.2 节, 经典条件作用的 TD 模型基本上是 semi-gradient-descent TD( ) 和线性函数近似算法。蒙塔古等人做了几个假设来建立这个比较。首先, 因为 TD 错误可以负但神经元不能有消极的射速, 他们认为多巴胺神经元活动的数量对应的是 t−1 + bt,bt 在哪里背景神经元的放电频率。一个消极的 TD 错误对应于多巴胺神经元的放电率低于其本底水平在每一个经典条件反射试验中，都需要第二个假设，以及它们如何被表示为学习算法的输入。这是我们在第 14.2.4 节中讨论的 TD 模型所讨论的问题。Montague 等人选择了一个完整的串行化合物 (CSC) 表示，如图 14.1 的左列所示，但是短时间内信号序列会持续到美国开始，这是一个非零奖励信号的到达。这种表现使得 TD 错误模拟了多巴胺神经元的活动不仅可以预测未来的奖励，而且它对预期的奖励何时到来也很敏感。必须有某种方法来跟踪感官提示和奖励到达之间的时间。如果刺激产生一系列内部信号，这些信号在刺激结束后继续存在，如果在刺激结束后的每个时间步上都有不同的信号，那么在刺激结束后的每个时间步上都有不同的状态。因此，与状态相关的 TD 错误可以对试验中的事件时间敏感。在这些关于背景放电率和输入表征的假设的模拟试验中，TD 模型的 TD 错误与多巴胺神经元的阶段活动非常相似。在下面 15.4 节中，我们对这些相似点的详细描述，TD 错误与多巴胺神经元活动的以下特征相平行:2) 在早期的学习中，奖励前的中性线索不会引起大量的阶段性多巴胺反应，但随着不断的学习，这些线索会获得预测价值，并引发阶段性多巴胺反应;3) 如果一个更早的线索可靠地先于一个已经获得了预测值的线索，阶段性的多巴胺反应就会转移到早的线索，停止后的线索;4) 如果在学习后忽略了预期的奖励事件，那么在奖励事件发生后不久，多巴胺神经元的反应就会低于其基线水平。虽然并不是每个多巴胺神经元都在舒尔茨和

2 TD 错误相关的文献中多巴胺神经元的活动, 他们 t 一样我们 t−1 = Rt + V(St)−V(圣−1)。

同事们在所有这些方面都表现得很好，大部分被监测的神经元的活动和 TD 的错误之间的惊人的对应关系为奖励预测错误假设提供了强有力的支持。然而，在某些情况下，基于假设的预测与实验中观察到的结果不相符。输入表征的选择对于 TD 错误与多巴胺神经元活动的某些细节，尤其是关于多巴胺神经元反应时间的细节匹配程度是至关重要的。我们在下面讨论了不同的想法，其中一些是关于输入表示和 TD 学习的其他特性，以使 TD 错误更适合数据，尽管主要的相似之处出现在了 Montague 等人使用的 CSC 表示中。总的来说，奖励预测错误假说在研究基于奖励的学习的神经科学家中得到了广泛的认可，并且在面对从神经科学实验中积累的结果时，它被证明是非常有弹性的。神经科学实验的准备我们的描述支持奖励预测错误的假设, 并提供一些背景, 这样假设的意义可以欣赏, 我们下一个礼物是什么了解多巴胺, 影响大脑结构, 以及它如何参与奖励学习。

14.4 多巴胺

多巴胺作为一种神经递质产生于哺乳动物中脑的神经元，它们的细胞主体主要位于哺乳动物中脑的两个神经元簇: 黑质 (SNpc) 和腹侧被盖区 (VTA)。在哺乳动物的大脑中，多巴胺在许多过程中扮演着重要的角色。其中最突出的是动机，学习，行动选择，大多数上瘾形式，精神分裂症和帕金森氏症。多巴胺被称为神经调节器，因为它除了直接快速刺激或抑制目标神经元外，还有许多功能。虽然关于多巴胺的功能和细胞效应的细节仍有很多未知，但很明显，它对哺乳动物大脑的奖赏处理是至关重要的。多巴胺并不是奖励过程中唯一的神经调节因子，它在厌恶情境中的作用——惩罚——仍然存在争议。多巴胺在非哺乳动物中也有不同的功能。但是没有人怀疑多巴胺对包括人类在内的哺乳动物的奖赏过程是必不可少的。一种早期的传统观点认为，多巴胺神经元向与学习和动机有关的多个大脑区域传播奖励信号。这一观点源自詹姆斯 • 奥兹和彼得 • 米尔纳在 1954 年发表的一篇著名论文，文中描述了电刺激对老鼠大脑某些区域的影响。他们发现，对特定区域的电刺激在控制老鼠行为方面起到了非常强大的奖励作用:“……”通过这种奖励，对动物行为的控制是极端的，可能超过了以前在动物实验

中使用过的任何其他奖励。”(Olds 和 Milner, 1954)。后来的研究表明，刺激在产生奖赏效应时最有效的部位会直接或间接地刺激多巴胺通路，而这些通路通常是由自然的奖赏刺激激发的。与这些相似的效果也在人体实验中被观察到。这些观察结果强烈地表明，多巴胺神经元的活动是奖励的信号。

但是，如果奖励预测错误假说是正确的——即使它只解释了多巴胺神经元活动的某些特征——这种对多巴胺神经元活动的传统看法并不完全正确: 多巴胺神经元的阶段性反应表明奖励预测错误，而不是奖励本身。在强化学习的术语中, 多巴胺神经元的相位的响应在时间 t 对应 t−1 = Rt + V(St)−V(圣 −1), 而不是 Rt。强化学习理论和算法有助于调和奖励预测错误观点与传统观念认为多巴胺信号的奖励。在我们讨论的很多算法在这本书中, 函数作为强化信号, 这意味着它是学习的主要原因。例如, 是经典条件作用的 TD 模型中的关键因素, 是强化学习的信号值函数和政策 actor-critic 架构 (章节 13.5 和 15.7)。Action-dependent 形式的 强化信号 q 学习和撒尔沙。奖励信号 Rt 是一个关键组成部分 t−1, 但它不是完整的行列式在这些算法的强化效果。额外的术语 V(St)−V(圣 −1) 的高阶强化部分 t−1, 即使奖励 (Rt ? = 0) 发生,TD 的错误可以沉默如果奖励是完全预测 (这是完全解释下面的 15.6 节)。仔细看看欧兹和米尔纳在 1954 年的论文，事实上，这篇论文主要是关于在仪器调节任务中电刺激的增强效果。电刺激不仅激发了老鼠的行为——通过多巴胺对动力的作用——还使老鼠迅速学会通过按压杠杆来刺激自己，而这是它们长时间经常做的事情。电刺激引起的多巴胺神经元活性增强了大鼠的杠杆按压。最近使用光遗传学方法的实验确定了多巴胺神经元的阶段反应作为强化信号的作用。这些方法使神经科学家能够精确地控制清醒行为的动物在一毫秒内所选择的神经元类型的活动。光遗传学方法将光敏蛋白引入到选定的神经元类型中，使这些神经元能够通过激光的闪烁来激活或沉默。第一个实验使用多巴胺神经元 optogenetic 方法研究表明,optogenetic 刺激生产阶段的多巴胺神经元激活小鼠足以条件老鼠喜欢的商会, 他们收到这刺激比室的另一边, 他们没有收到, 或低的频率, 刺激 (蔡 et al . 2009 年)。在另一个例子中，Steinberg 等人 (2013 年) 利用多巴胺神经元的光遗传学激活，在预期有奖励刺激但在正常的多巴胺神经元活动停止时，在大鼠中制造人为的多巴胺神经元活动爆发。当这些停顿被人为的爆发所取代时，当响应通常因缺乏强化而减少时 (在绝灭试验中)，当响应通常因预期的奖励而被阻断时 (阻塞范式;14.2.1 节)。额外的多巴胺的增强功能的证据来自 optogenetic 与果蝇实验, 除了这些动物多巴胺在哺乳动物的效果是相反的效果: 光引发多巴胺神经元活动的行为就像电动脚冲击强化回避行为, 至少在人口

多巴胺神经元的活化 (clarichang - chang et al. 2009)。尽管这些 optogenetic 实验表明, 相位的多巴胺神经元活动特别像一个 TD 错误, 他们令人信服地表明, 相位的多巴胺神经元活动行为就像 行为 (或者像- 在果蝇行为) 作为预测算法的强化信号 (经典条件作用) 和控制 (工具性条件作用)。

一个神经元的轴突轴位轴突多巴胺是一种神经递质。这些轴突与目标神经元的大量树突产生突触联系大脑区域。改编自《神经科学杂志》,《松田》、《芙蓉》、《中村》、《平木》、《富士山》、《天宫》，2009 年第 29 卷，第 451 页。多巴胺神经元特别适合向大脑的许多区域传播强化信号。这些神经元有巨大的轴突，每一个都释放出比典型神经元的轴突多 100 到 1000 倍的多巴胺。SNpc 或 VTA 多巴胺神经元的每一个轴突都会在目标大脑区域的神经元树突上产生大约 50 万个突触联系。如果多巴胺神经元广播一个强化信号像强化学习的 , 因为这是一个标量信号, 即。一个数字，SNpc 和 VTA 中的所有多巴胺神经元或多或少都将被期望以相同的方式激活，这样它们就会在几乎同步的情况下，将相同的信号发送到它们的轴突目标的所有位置。尽管人们普遍认为多巴胺神经元确实是这样共同作用的，但现代证据表明，更复杂的情况是，不同的多巴胺神经元亚群对输入的反应不同，这取决于它们发出信号的结构和不同的结构这些信号作用于目标结构的方式。多巴胺除了发出信号外，还具有其他功能，即使是多巴胺神经元也有信号 RPEs，根据这些结构在产生强化行为中的作用，将不同的 RPEs 发送到不同的结构是有意义的。这是超出了我们对待任何细节在这本书中, 但向量值 RPE 信号意义从强化学习的角度来看, 当决策可以分解成单独的 sub-decisions, 或更普遍的是, 作为一种解决结构版本的信贷分配问题: 你如何分配信贷成功 (或失败) 负责决定在众多组件的结

构可能是参与生产吗? 我们将在下面的第 15.10 节中对此进行更多的讨论。大多数多巴胺神经元的轴突与额叶皮层和基底神经节的神经元产生突触联系，大脑的区域参与自发运动、决策、学习和计划等认知功能。因为大多数的想法有关

多巴胺增强学习集中在基底神经节，而来自多巴胺神经元的连接在那里特别密集，我们集中在基底神经节。基底神经节是位于前脑基底部的一组神经元群，或核。基底神经节的主要输入结构称为纹状体。基本上所有的大脑皮层和其他结构都为纹状体提供输入。皮层神经元的活动传递了丰富的有关感觉输入、内部状态和运动活动的信息。皮层神经元的轴突在纹状体的主要输入/输出神经元的树突上形成突触联系，称为中刺神经元。纹状体的输出通过其他基底节核和丘脑回至皮质额区和运动区，使纹状体有可能影响运动、抽象决策过程和奖赏处理。纹状体的两个主要分支对强化学习很重要: 背侧纹状体，主要涉及影响行为选择; 腹侧纹状体，被认为是奖励处理的不同方面的关键，包括情感对感觉的分配。中等大小的刺状神经元的树突上覆盖着刺状突起，其顶端的神经元轴突与突触有接触。同时，与这些脊髓的突触接触——在这种情况下，是多巴胺神经元的轴突 (图 15.1)。这种排列将皮质神经元的突触前活动聚集在一起，

图 15.1: 纹状体神经元的脊柱，显示皮层和多巴胺神经元的输入。皮层神经元的轴突通过皮质纹状突触释放神经递质谷氨酸来影响纹状体神经元。VTA 或 SNpc 多巴胺神经元的轴突通过脊柱 (右下方)。这个轴突上的“多巴胺静脉曲张”会在脊柱茎处或附近释放多巴胺，这种排列将突触前的输入，突触后的活动聚集在一起纹状神经元和多巴胺，这使得几种学习规则成为可能层次的突触的可塑性。多巴胺神经元的每一个轴突都与大约 50 万个脊柱的茎产生突触接触。我们讨论中忽略的一些复杂性可以通过其他神经递质途径和多种受体类型来体现，比如 D1 和 D2 多巴胺受体，多巴胺可以在棘突和其他突触后部位产生不同的效果。《神经生理学杂志》，W. 舒尔茨，1998 年第 80 页，第 10 页。

中棘神经元突触后活动，多巴胺神经元的输入。这些刺的实际情况是复杂的，并不完全了解。图 15.1 显示了两种类型的多巴胺受体，谷氨酸受体——皮层输入的神经递质——以及各种信号相互作用的多种方式，暗示了复杂性。但越来越多的证据表明，从大脑皮层到大脑纹状体 (神经科学家称其为皮质纹状体突触) 的神经突触效率的变化，在很大程度上依赖于适当的多巴胺信号。

14.5 奖励预测误差假设的实验支持

多巴胺神经元对强烈的、新奇的或意想不到的视觉和听觉刺激做出反应，这些刺激会触发眼睛和身体的运动，但它们的活动与运动本身几乎没有关系。这是令人惊讶的，因为多巴胺神经元的退化是帕金森病的一个原因，其症状包括运动障碍，尤其是自我启动的运动障碍。Romo 和 Schultz(1990)、Schultz 和 Romo(1990) 和 Romo(1990) 在多巴胺神经元活动与刺激引发的眼睛和身体运动之间的弱关系的驱使下，通过记录多巴胺神经元的活动和猴子移动手臂时的肌肉活动，迈出了实现奖励预测错误假说的第一步。他们训练两只猴子把手从静止的位置伸到一个装着苹果、饼干或葡萄干的箱子里，这时猴子看到并听到箱子的门开了。猴子可以抓住食物，把食物送到嘴里。当一只猴子在这方面做得很好之后，它被训练做另外两项任务。第一个任务的目的是观察当运动自我启动时，多巴胺神经元会做什么。箱子是开着的，但上面盖着盖子，猴子看不见里面，但可以从下面伸进去。没有任何触发刺激出现，当猴子伸手去拿食物吃后，实验者通常 (虽然不是总是)，安静地，不被猴子看见，把食物粘在坚硬的铁丝上，取代食物在箱子里。在这里，Romo 和 Schultz 观察到的多巴胺神经元的活动与猴子的动作没有关系，但是当猴子第一次接触食物时，这些神经元中的很大一部分会产生相对应的反应。当猴子在没有食物的情况下触摸金属丝或探索垃圾箱时，这些神经元没有反应。这很好地证明了神经元对食物的反应，而不是对任务的其他方面的反应。Romo 和 Schultz 的第二个任务的目的是观察当运动被刺激物触发时会发生什么。这项任务使用了一个不同的箱子和一个可移动的盖子。打开垃圾箱的景象和声音引发了人们对垃圾箱的移动。在这种情况下，

Romo 和 Schultz 发现在经过一段时间的训练后，多巴胺神经元不再对食物的触摸做出反应，而是对食物箱的打开盖的视觉和声音做出反应。这些神经元的阶段性反应已经从奖励本身转移到刺激来预测奖励的有效性。在随后的一项研究中，Romo 和 Schultz 发现了大部分多巴胺神经元的活动

在行为任务之外，被监控的人对打开垃圾箱的视觉和声音没有反应。这些观察结果表明，多巴胺神经元对运动的开始和刺激的感官特性都没有反应，而是在发出奖赏的信号。舒尔茨的团队进行了许多涉及 SNpc 和 VTA 多巴胺神经元的额外研究。一组特定的实验结果表明，多巴胺神经元的相对应于 TD 误差，而不是像 Rescorla-Wagner 模型 (14.3) 那样的更简单的错误。在第一个实验中 (Ljungberg, Apicella，和 Schultz, 1992)，猴子们被训练在灯光作为“触发信号”被照亮以获得一滴苹果汁后，去压低杠杆。正如 Romo 和 Schultz 早些时候所观察到的，许多多巴胺神经元最初对奖励做出反应——果汁的下降 (图 15.2，顶部面板)。但是，随着训练的继续，这些神经元中的许多失去了奖赏反应，取而代之的是对预示奖赏的光线的反应 (图 15.2，中间面板)。随着持续的训练，杠杆按压变得更快，而对触发信号做出反应的多巴胺神经元数量减少。在这项研究之后，同样的猴子接受了新任务的训练 (Schultz, Apicella，和 Ljungberg, 1993)。在这里，猴子面对着两个杠杆，每个杠杆上面都有一盏灯。其中一盏灯的照明是一个“指令提示”，指示两个杠杆中的哪一个

图 15.2: 多巴胺神经元的反应由最初的反应转变为最初的奖赏。早期预测刺激。这些是被监测的多巴胺神经元在小时间间隔内产生的动作电位的数量的图，在所有被监测的多巴胺神经元 (这些数据从 23 个神经元到 44 个神经元) 上取平均数。图中: 未预料到的苹果汁会刺激多巴胺神经元。中间: 学习, 多巴胺神经元对预测奖励的触发线索产生反应，对奖励的传递失去反应。底部: 在触发信号前增加一个指令提示 1 秒，多巴胺神经元将他们的反应从触发信号转移到之前的指令提示。舒尔茨等人 (1995)，麻省理工学院出版社。

14.6 奖励预测误差假设 389 的实验支持

会产生一滴苹果汁。在此任务中，指令提示在前一个任务的触发提示之前以 1 秒的固定间隔进行。猴子们学会了在看到触发信号之前不去触碰，多巴胺神经元的活动增加了，但是现在被监测的多巴胺神经元的反应几乎只发生在早期的指令信号上，而不是触发信号 (图 15.2，底部面板)。在这里，当这个任务被很好地学习之后，对指令信号做出反应的多巴胺神经元的数量又大大减少了。在完成这些任务的学习过程中，多巴胺神经元的活动从最初对奖励的反应转变为对早期的预测刺激的反应，首先是触发刺激，然后是更早的指示信号。re -

图 15.3: 在预期的时间后不久，多巴胺神经元的反应低于基线奖励失败发生。图中: 一滴苹果汁意想不到的分泌会激活多巴胺神经元。中间: 多巴胺神经元对条件刺激 (CS) 做出反应，它可以预测奖励，但对奖励本身没有反应。底部: 当 CS 预测的奖励没有发生时，多巴胺神经元的活动在预期奖励发生后不久就会低于基线。在每个面板的顶部显示平均值被监测的多巴胺神经元在指定时间间隔内产生的动作电位的数量。下面的光栅图显示了被监控的单个多巴胺神经元的活动模式; 每个点代表一个动作电位。1997 年 3 月 14 日，来自舒尔茨、大安和蒙太古，预测与奖励的神经基板，科学，第 275 卷，第 5306 期，第 1593-1598 页。经美国科学促进会许可转载。随着时间的推移，它从后来的刺激中消失了。这种对早期奖励预测者的反应转移，而对后来的预测者失去反应是 TD 学习的一个标志 (例如，参见图 14.2)。刚刚描述的任务揭示了多巴胺神经元活动与 TD 学习共享的另一个特性。猴子有时按错了钥匙，也就是说，钥匙除了指示的钥匙外，没有得到任何奖励。在这些试验中，许多多巴胺神经元在奖励通常的交付时间后不久，其放电率急剧下降，低于基线，而这是在没有任何外部提示来标记奖励通常的交付时间的情况下发生的 (图 15.3)。不知怎么的

猴子们在体内追踪奖励的时间。(反应时间是 TD 学习的最简单版本需要修改的一个领域，以解释多巴胺神经元反应时间的一些细节。我们将在下一节中讨论这个问题。上述研究的观察结果让舒尔茨和他的团队得出结论，多巴胺神经元对未预料到的奖励、对奖励最早的预测

因子做出反应，如果奖励或奖励的预测因子在预期时间内没有出现，那么多巴胺神经元的活动就会低于基线水平。熟悉增强学习的研究人员很快认识到，这些结果与 TD 错误在 TD 算法中的增强信号表现非常相似。下一节通过详细分析一个特定示例来研究这种相似性。

14.7 TD 错误/多巴胺的信件

本节解释之间的通信 TD 误差 和多巴胺神经元的阶段反应在刚刚描述的实验观察。我们检查 如何变化的学习任务就像上面描述的一个猴子第一次见到一个指令信号和一个固定的时间后必须正确应对触发提示为了获得奖励。我们使用这个任务的一个简单的理想化版本，但是我们比平常更详细，因为我们想强调 TD 错误和多巴胺神经元活动平行的理论基础。第一个简化的假设是，代理已经学会了获得奖励所需的行为。然后，它的任务就是精确地预测它所经历的状态的未来回报。然后，这是一个预测任务，或者更严格地说，是一个策略评估任务: 学习一个固定策略的值函数 (第 4.1 和 6.1 节)。要学习的值函数为每个状态分配一个值，该值预测如果代理根据给定的策略选择动作，那么返回值将跟随该状态的返回值，其中返回值是所有未来奖励的 (可能折现的) 总和。这是不现实的, 因为猴子的情况, 因为猴子的模型可能会在同一时间学习这些预测, 它是学习正确采取行动 (强化学习算法, 学习政策以及价值功能, 比如 actor-critic 算法), 但这种情况下比一个简单的描述一个政策和值函数同时学习。现在假设代理的经验分为多个试验，在每个试验中，相同的状态序列重复，在试验的每个时间步上都有不同的状态。进一步假设所预测的回报仅限于试验的回报，这使得试验类似于我们定义的强化学习。当然，在现实中，预测的回报并不局限于单一的试验，试验之间的时间间隔是决定动物学习什么的一个重要因素。这也适用于 TD 学习，但这里我们假设回报不会在多次试验中累积。考虑到这一点，像舒尔茨和他的同事所做的实验就相当于强化学习。(虽然在这个讨论中，我们将使用学期试验而不是插曲来更好地与实验相关。)

和往常一样，我们还需要对状态如何表示为学习算法的输入做出一个假设，这个假设会影响道明错误与多巴胺神经元活动的密切程度。我们稍后将讨论这个问题，但目前我们假设与 Montague 等人 (1996) 所使用的 CSC 表示方式相同，即在试验的每个步骤中，每个国家都有一个单独的内部刺激。这将过程简化为本书第一部分所涉及的表格情况。最后，我们假定代理使用 TD(0) 来学习一个值函数，V，存储在一个初始化为所有状态为零的查找表中。我们也认为这是一个确定的任务和折现系数, , 几乎是一个, 这样我们可以忽略它。图 15.4 显示的时间课程 R,V, 在几个阶段的学习在这个政策评估的任务。时间轴表示在试验中访问状态序列的时间间隔 (为了清晰起见，我们省略了显示单个状态)。在每次试验中，奖励信号为零，除非当代理到达奖励状态，在时间线的右端，当奖励信号变成一个正数时，R?TD 学习的目标是预测在试验中访问的每个州的回报，在这个未折现的案例中，考虑到我们假设预测仅限于个别试验，这仅仅是 R 吗? 为每个状态。t R ?

R 常规的预测因素 R 在这个区间 V R

图 15.4:TD 的行为错误 在 TD 学习阶段的特点是一致的多巴胺神经元的激活。(是 TD 错误可以在时间 t, 即。, t−1)。顶部: 一个状态序列，以一个正则预测符的区间表示，后面跟着一个非零的奖励 R? 早期的学习: 初始值函数,V, 和最初的 , 起初等于 R ?。学习完成: 价值函数准确地预测未来的回报, 是积极的最早预测状态, = 0 时的非零回报。R ? 省略了: 当时预测省略奖励, 变得消极。有关发生这种情况的完整解释，请参见文本。

奖励状态之前是奖励预测状态的序列，最早的奖励预测状态显示在时间线的左端。这就像在试验开始时的状态，例如，在舒尔茨等人 (1993) 的猴子实验中，用指令提示标记的状态。这是试验中第一个可靠地预测试验结果的状态。(当然，在现实中，在之前的试验中访问的状态甚至是更早的奖励预测状态，但是因为我们将预测限制在单独的试验中，这些不能作为试验回报的预测者。下面我们给出一个更令人满意的，尽管更抽象的描述，一个最早的奖励预测的状态。在试验中，最新的奖励预测状态是试验开始前的状态。这是图 15.4 中时间线最右端附近的状态。值得注意的是，试验的回报状态并不会预测试验的回报: 这个状态的价值将会

在接下来的所有试验中预测试验的回报，这里我们假设在这个情景式中是零。图 15.4 显示了初审时间课程 V 和 的图形标记的早期学习。“因为整个实验中奖励信号都是零，除了达到奖励状态时，所有 V 值都是零，所以 TD 误差也为零，直到变成 R?”在有益的状态。在此之前因为 t−1 = Rt + Vt−Vt Rt + 0−−1 = 0 = Rt, 哪个是零, 直到它等于 R ? 当奖励。Vt 和 Vt−1 分别估算值的州访问有时 t,t−1 审判。这个学习阶段的 TD 错误类似于训练开始时的多巴胺神经元对未预料到的奖励 (如一滴苹果汁) 的反应。在第一次试验和所有后续试验中，TD(0) 更新发生在每个状态转换上，如第 6 章所述。这连续地增加了奖励预测状态的值，增加了从奖励状态向后扩展的值，直到值收敛到正确的返回预测。在这种情况下 (因为我们假设没有折现) 正确的预测等于 R? 对于所有的奖励预测状态。这可以在图 15.4 中看到 V 标记为“学习完成”的图形，其中从最早到最近的奖励预测状态的所有值都等于 R? 早期奖励预测状态之前的状态的值仍然很低 (图 15.4 显示为零)，因为它们不是奖励的可靠预测者。当学习完成时，即当 V 达到其正确值时，与任何奖赏预测状态的转换相关的 TD 错误为零，因为预测现在是准确的。这是因为从 reward-predicting 状态过渡到另一个 reward-predicting 状态, 我们有 t Rt + Vt Vt−−−1 = 1 = 0 + R ?−R ?= 0, 从最新的 reward-predicting 状态转换到有益的状态, 我们有 t Rt + Vt Vt−−−1 = 1 = R ? + 0−R ?= 0。另一方面，从任何状态到最早的奖励预测状态的 TD 误差是正的，因为这个状态的低值与后面奖励预测状态的大值不匹配。事实上, 如果一个国家的价值之前最早 reward-predicting 状态是零, 然后转换到最早的 reward-predicting 状态后, 我们会 t Rt + Vt Vt−−−1 = 1 = 0 + R ?−0 = R ?。 的学习完成的图在图 15.4 显示了这个正值最早 reward-predicting 状态,0 和其他地方。

在过渡到最早的奖励预测状态时，TD 阳性错误类似于多巴胺对最早的奖励刺激的持续反应。同样，当学习完成时，从最新的奖励预测状态到奖励状态的转换会产生零 TD 错误，因为最新的奖励预测状态的值如果正确，就会抵消奖励。与此类似的观察是，对完全预测的奖励而不是未预测的奖励，产生阶段性反应的多巴胺神经元更少。学习后, 如果奖励是突然省略,TD 的错误在老时间去负的奖励, 因为最新的价值 reward-predicting 状态然后过高: t Rt + Vt Vt−−−1 = 1 = 0 + 0−R ?=−R ? 如图所示, 右端 R 省略的图的如图 15.4 所示。这就像在舒尔茨等人 (1993) 的实验中所看到的，在预期的奖励被忽略时，多巴胺神经元的活动在低于基线的情况下减少，如图 15.3 所示。最早的奖励预测状态的想法值得更多的关注。在上面描述的场景中，由于经验被划分为试验，我们假设预测仅限于单个试验，所以最早的回报预测状态总是试验的第一状态。很明显, 这是人为的。一种更普遍的方式来考虑最早的奖励预测状态，那就是它是一个无法预测的奖励预测，而且可能有很多这样的状态。在动物的一生中，许多不同的状态可能先于最早的奖赏预测状态。然而，由于这些国家往往被其他不预测回报的国家所取代，它们的回报预测能力，也就是它们的价值，仍然很低。一个 TD 算法，如果在动物的整个生命中运行，也会更新这些状态的值，但是更新不会持续积累，因为，根据假设，这些状态都不会可靠地先于最早的奖励预测状态。如果他们中的任何一个这样做了，他们也将是回报预测国家。这也许可以解释为什么在过度训练的情况下，多巴胺的反应会减少到甚至是在试验中最早的奖励预测刺激。在过度训练的情况下，人们可能会认为，即使是一个没有预测的预测状态，也会通过与早期状态相关的刺激来预测: 动物在实验任务内外与环境的相互作用会变得很平常。然而，当你打破常规，开始一项新的任务时，你会看到 TD 错误再次出现，就像在多巴胺神经元活动中观察到的那样。上面描述的示例解释了为什么当动物在类似于我们的示例中的理想化任务中学习时，TD 错误与多巴胺神经元的阶段活动具有相同的关键特征。但不是每个属性的相位的多巴胺神经元的活动一致所以整齐 的属性。最令人不安的差异之一涉及奖励比预期更早发生时发生的情况。我们已经看到，忽略一个预期的奖励会在奖励的预期时间产生一个消极的预测错误，这对应于当这发生时多巴胺神经元的活动低于基线下降。如果奖励比预期来得晚，那么这就是一个意外的奖励，并产生一个积极的预测错误。这发生在 TD 错误和多巴胺神经元反应。但是当奖赏比预期来得更早时，多巴胺神经元就不会做 TD 错误所做的事情——至少是蒙塔古等人 (1996) 和我们在我们的例子中使用的 CSC 表示法。多巴胺

神经元确实会对早期的奖励做出反应，这与阳性的 TD 错误是一致的，因为那时奖励不会

被预测出来。然而，当预期的奖励被忽略时，TD 的错误是消极的，而与此相反，在 TD 模型预测的方式下，多巴胺神经元的活动并没有低于基线 (Hollerman 和 Schultz, 1998)。动物大脑中正在发生的事情比单纯用 CSC 表示学习 TD 要复杂得多。通过选择适用于 TD 算法的参数值，并使用除 CSC 表示之外的刺激表示，可以解决 TD 错误和多巴胺神经元活动之间的一些不匹配。例如，为了解决刚才描述的早期奖励不匹配问题，Suri 和 Schultz(1999) 提出了 CSC 表示，在 CSC 表示中，早期刺激引发的内部信号序列被奖励的发生所抵消。Daw、Courville 和 Touretzky(2006) 提出的另一个建议是，大脑的 TD 系统使用在感觉皮层进行的统计建模所产生的表征，而不是基于原始感觉输入的简单表征。Ludvig、Sutton 和 Kehoe(2008) 发现，与使用微刺激 (MS) 表征的 TD 学习 (图 14.1) 比使用 CSC 表征更适合早期奖励和其他情况下多巴胺神经元的活动。Pan、Schmidt、Wickens 和 Hyland(2005) 发现，即使有 CSC 表示法，长时间的合格性跟踪也能提高 TD 误差在多巴胺神经元活动某些方面的适应性。一般来说，许多关于 TD-error 行为的细节依赖于资格跟踪、贴现和刺激表示之间的微妙交互。诸如此类的发现详细阐述并完善了奖励预测错误假说，但并没有驳斥其核心观点，即多巴胺神经元的阶段性活动具有明显的 TD 性错误信号。另一方面, 还有其他 TD 理论和实验数据之间的差异不是那么容易适应通过选择参数值和刺激表示这些差异的 (我们提到一些书目的和历史的备注部分结束时, 这一章), 和更多的不匹配可能会发现为神经科学家进行更加精炼实验。但是奖励预测错误假说已经非常有效地作为一种催化剂来提高我们对大脑奖励系统如何工作的理解。复杂的实验被设计来验证或反驳来自假设的预测，而实验结果反过来又导致了 TD 误差/多巴胺假设的细化和细化。这些发展的一个显著的方面是，强化学习算法和理论与多巴胺系统的特性紧密相连，从一个计算的角度出发，完全没有关于多巴胺神经的相关特性的知识——记住，TD 学习及其与最优控制和动态规划之间的联系是在许多年之前发展起来的。除了考虑到多巴胺神经元阶段活动的许多特征外，奖励预测错误假说还将神经科学与强化的其他方面联系起来

特别是学习使用 TD 错误作为强化信号的算法。神经科学还远未达到完成对电路的理解, 分子机制, 相位的多巴胺神经元的活动和功能, 但证据支持奖励预测误差假设, 随着证据表明, 相位的多巴胺反应是学习的强化信号, 表明大脑可能实现类似的 actor-critic 算法 TD 错误扮演至关重要的角色。其他的强化学习算法也是可行的，但是行为-批判算法特别适合哺乳动物大脑的解剖和生理学，正如我们在下面两个部分中所描述的。

14.8 神经 Actor-Critic

行为批评算法学习策略和价值函数。“演员”是学习策略的组成部分，而“批评家”则是学习任何政策的成分，以“批评”演员的行为选择。批评家使用 TD 算法来学习参与者当前策略的状态值函数。价值函数允许评论家批评演员的行动选择通过发送 TD 错误, , 演员。积极的 意味着行动是“好”, 因为它导致了国家好于预期值; 负 意味着行动是“坏”, 因为它导致了一个国家与一个弱于预期值。根据这些评论，该演员不断更新其政策。行为批评算法的两个显著特点是，大脑可能会实现这样的算法。首先，表演-批评家算法的两个组成部分——行动者和评论家——表明，条纹的两个部分——背侧和腹侧细分 (15.4 节)——对于基于奖励的学习来说都是至关重要的——可能分别起着类似演员和评论家的作用。actor - critics 算法的第二个特性是，虽然 TD 错误对每个部分的学习都有不同的影响，但对于行动者和评论者来说，TD 错误同时具有增强信号的双重作用。这与神经回路的几个特性非常吻合: 多巴胺神经元的轴突瞄准纹状体的背侧和腹侧的亚区; 多巴胺似乎对调节两种结构的突触可塑性至关重要; 神经调节者，如多巴胺在目标结构上的作用取决于目标结构的性质，而不仅仅取决于神经调节因子的性质。第 13.5 节提出了作为策略梯度方法的 actor -批评家算法，但是 Barto、Sutton 和 Anderson(1983) 的 actor -批评家算法更简单，作为一个人工神经网络 (ANN) 提出。这里我们描述了一种类似 Barto 等人的 ANN 实现，我们跟随 Takahashi, Schoenbaum 和 Niv(2008) 给出了一个关于这个 ANN 如何被大脑中的真实神经网络实现的示意图。我们

把关于演员和评论家学习规则的讨论推迟到第 15.8 节，我们将它们作为政策梯度公式的特殊案例，讨论他们对多巴胺如何调节突触可塑性的建议。

图 15.5a 展示了一种作为 ANN 的 actor -批评家算法的实现，该算法使用组件网络实现了参与者和批评者。评论家包含单个 neuron-like 单元,V, 输出活动代表状态的值, 一个组件显示为钻石标记 TD 计算 TD 错误结合 V 的输出与奖赏信号, 与先前的状态值 (所显示循环从 TD 钻石本身)。行动者网络有一层 k 行动者单位标记为 Ai, i = 1，…每个动作单元的输出都是 k 维动作向量的一个分量。另一种选择是有 k 个单独的动作，一个由每个行动者单位命令，它们相互竞争要被执行，但是在这里我们将把整个 a 向量看作一个动作。

图 15.5: 角色评论家 ANN 和假设的神经实现。a) 作为 ANN 的 actor - critics 算法。演员调整政策基于 TD 它收到的错误评论家; 使用相同的 评论家调整州值参数。批评家从奖励信号 R 和当前状态值估计的变化中产生一个 TD 错误。演员是没有直接访问奖励信号，评论家也没有直接访问行为。b) 假定的行为-批评算法的神经实现。行动者和批判者的价值学习部分分别被置于纹状体的背侧和腹侧亚区。在 VTA 和 SNpc 上的多巴胺神经元传递 TD 的错误，以调节从皮质区输入到腹侧和背纹状体的突触效果的变化。从神经科学的前沿，vol. 2(1)，2008, Y. Takahashi, G。舒恩鲍姆 (Schoenbaum) 和 Y. Niv，让批评者闭嘴: 了解可卡因敏感化的影响在行动者/评论家模型的背景下的背外侧和腹侧纹状体。

批评家和参与者网络都接收由代表代理环境状态的多个特性组成的输入。(请参阅第 1 章，强化学习因子的环境包括包含强化学习因子的“有机体”内部和外部的组件。) 图中显示了标记为 x1, x2 的圆…，xn，显示两次以保持图形简单。代表突触效能的权重与每个特征 xi 与批评家单元 V 以及每个动作单元 Ai 之间的连接有关。评论家网络中的权重参数化值函数，参与者网络中的权重参数化策略。随着这些权重的变化，网络学习根据我们在下一节中描述的批评家和演员学习规则而变化。评论家的电路产生的 TD 误差是评论家和行动者网络中改变权重的增强信号。如图 15.5 的线标记“TD 误差 ”扩展所有的评论家和演员网络连接。这方面的网络实现, 一起奖励预测误差假设和多巴胺神经元的活动这一事实是如此广泛分布广泛的这些神经元轴突乔木, 表明一个 actor-critic 网络这样可能不会太牵强的作为一个假说犒赏学习如何在大脑中可能发生的。图 15.5b 建议——非常示意图——根据 Takahashi 等人 (2008) 的假设，图中左边的 ANN 如何映射到大脑的结构上。假设将行动者和批判者的价值学习部分分别置于纹状体的背侧和腹侧亚区，即基底神经节的输入结构中。回顾第 15.4 节，背纹状体主要涉及影响行为选择，腹纹状体被认为是奖励处理的不同方面的关键，包括情感价值分配给感觉。大脑皮层和其他结构一起向纹状体发送信息，传递有关刺激、内部状态和运动活动的信息。在这个假设的 actor -评论大脑的实施过程中，腹侧纹状体向 VTA 和 SNpc 发送价值信息，在这些信息中，这些核中的多巴胺神经元将其与奖励的信息相结合，从而产生与 TD 错误相关的活动 (尽管多巴胺能神经元究竟如何计算这些错误，目前还不清楚)。TD 误差 的线在图 15.5 成为标记在图 15.5 b“多巴胺”, 代表了广泛的分支轴突在腹侧被盖区多巴胺神经元的细胞体,SNpc。参考图 15.1，这些轴突使突触与中刺神经元的树突相连，这是纹状体背侧和腹侧部的主要输入/输出神经元。向纹状体发送输入的皮层神经元的轴突在这些刺的顶端产生突触接触。根据假设，在这些脊柱中，从皮质区到纹状体的突触的功效的变化受学习规则的支配，这是由多巴胺提供的强化信号决定的。图 15.5b 所示的假设的一个重要含义是，多巴胺信号不是像增强学习的标量 Rt 那样的“主”奖励信号。事实上，这个假设意味着，一个人不一定能够探测到大脑，并记录任何单个神经元活动中的信号。

许多相互关联的神经系统产生与奖励相关的信息，根据不同的奖励类型，不同的结构被招募。多巴胺神经元从许多不同的大脑区域接收信息，所以输入到 SNpc 和 VTA，在图 15.5b 中标记为“奖励”的输入应该被认为是奖励相关信息的矢量，沿着多个输入通道到达这些细胞核的神经元。那么，理论上的标量奖励信号 Rt 可能对应的是，所有与奖励相关的信息对多巴胺神经元活动的净贡献。这是大脑不同区域的许多神经元活动模式的结果。虽然图 15.5b 所示的行为-批判神经实现在某些方面可能是正确的，但它显然需要被细化、扩展和修改，才能成为一个完整的多巴胺神经元的阶段活动功能模型。本章末尾的历史和文献注释部分引用了一些出版物，这些出版物更详细地讨论了这一假设的经验支持和不足之处。我们现在详细

地看看行动者和批评家学习算法对控制皮质纹状体突触效率变化的规则的建议。

14.9 演员和评论家的学习规则

如果大脑并实现类似 actor-critic 算法和假设的数量多巴胺神经元广播一个常见的强化信号 corti-costriatal 突触的背侧和腹侧纹状体如图 15.5 b(这可能是一个简化我们上面提到的), 然后这个强化信号以不同的方式影响这两个的突触结构。学习规则的评论家和演员使用相同的强化信号,TD 误差 , 但其对学习的影响是不同的这两个组件。TD 错误 (结合资格跟踪) 告诉参与者如何更新操作概率，以达到高值状态。学习的演员就像使用 Law-of-Effect-type 工具性条件作用学习规则 (1.7 节): 演员保持 尽可能积极的工作。另一方面，TD 错误 (与合格跟踪结合) 告诉评论家改变值函数参数的方向和幅度，以提高其预测精度。评论家的作品减少 的大小尽可能接近于零使用学习规则的 TD 模型经典条件作用 (14.2 节)。批评家和演员学习规则之间的差异是相对简单的，但是这种差异对学习有着深远的影响，对于演员-批评家算法的工作方式是至关重要的。区别仅仅在于每种学习规则使用的资格跟踪。一套以上的学习规则可以在像图 15.5b 这样的神经网络中使用，但是，具体地说，这里我们关注的是在 13.6 节中提出的符合资格的跟踪问题的 actor -批评家算法。在每一个从国家圣状态转换圣 + 1, 采取行动和接收行动 Rt + 1, 这种算法计算 TD 错误 ( ), 然后更新资格跟踪向量 (zw t 和 z t)

评论家和演员的参数 (w 和), 根据

t = Rt v̂+ 1 + (圣 + 1 w)−v̂(St,w),zw t = wzw t−1 + v̂(St,w), z t = z −1 + ln (| 圣,),w←w + w tzw t, ← + z t, 在 (0,1) 是一个折现率参数, wc [0,1], wa [0,1]bootstrap-ping 参数评论家和演员分别和 w > 0 和 > 0 类似步长参数。认为近似值函数 v̂ 单个线性 neuron-like 单元的输出, 称为评论家单位和标记在图 15.5 v。值函数是状态 s、x(s) = (x1(s) 的特征向量表示的线性函数⋯,xn(s))?，由权向量 w = (w1) 参数化。wn)?:

v̂(s,w)= w ? x(s)。(15.1) 每一个 xi(s) 就像神经元突触前信号，其效力为 wi。批评家的重量增加根据 w tzw t, 上述规则的强化信号, t, 对应于一个多巴胺信号被广播给所有的评论家的突触。资格跟踪矢量,zw t, 评论家单位的跟踪 (平均最近的值) v̂(St,w)。因为 v̂(s,w) 是线性权重, v̂(St,w)= x(St)。从神经学的角度来说，这意味着每个突触都有自己的合格痕迹，这是向量 zw t 的一个组成部分。突触的合格性痕迹是根据到达突触的活动水平累积的，即突触前活动水平，这里由到达突触的特征向量 x(St) 的组成部分表示。跟踪否则衰减到零速度由分数 w。只要符合条件的跟踪不为零，突触就可以进行修改。突触的效能是如何改变的，取决于突触到达的强化信号，而突触是合格的。我们称之为合格的痕迹就像这些批评家单位的突触的不偶然的合格的痕迹因为它们只依赖于突触前的活动而不依赖于突触后的活动。批评家单位的突触的非偶然资格痕迹意味着评论家单位的学习规则实质上是第 14.2 节中描述的经典条件作用的 TD 模型。根据我们对批评家单元的定义和它的学习规律，图 15.5a 的批评家和 Barto 等人 (1983) 的《ANN》的批评家是一样的。显然，像这样的批评家只包含一个线性神经元单元是最简单的起点; 这个批评家单元是一个更复杂的神经网络的代理，它能够学习更复杂的值函数。图 15.5a 中的演员是 k 个神经元样的演员单元的单层网络，每一个接收的特征向量 x(St) 都是由评论家单元接收的。每个行动者单位 j, j = 1，⋯k 有自己的权向量, j, 但因为演员单位都是相同的, 我们单位和省略的描述只有一个下标。一种方法为这些

按照上述方程中给出的行为-批判算法，每一个单位都是伯努利-逻辑单元。这意味着每个参与者单元的每次输出都是一个随机变量，取值为 0 或 1。把值 1 想象成神经元放电，也就是说，释放一个动作电位。加权和, ? x(St), 一个单元的输入向量决定了单位的行动通过指数 soft-max 概率分布 (13.2), 这两个动作是物流功能:

(1 | s)= 1− (0 | 年代,)= 1 1 + exp(− ? x(s))。(15.2) 每个演员的重量单位是递增, 如上所述, 通过: ← + tz t, 在 又对应于多巴胺信号: 相同的强化信号, 发送到所有的评论家的突触。图 15.5 显示了 t 被广播给所有的突触演员单位 (这使得这个演员网络团队强化学习代理, 下面我们在 15.10 节讨论)。演员资格跟踪矢量 z t 是一个跟踪最近的值) 的 (平均 ln (| 圣,)。

要理解此资格跟踪，请参考第 13.5 条，它定义了这类单元，并要求您为其提供学习规则。锻炼让你表达 ln (| 年代,) 的,x(s), (| 年代,)(任意状态和行动) 通过计算梯度。对于 t 时刻实际发生的动作和状态，答案是

(| 圣,)= ? 在 − (| 圣,)? x(St)。(15.3) 与只积累突触前活动 x(St) 的批评家突触的非偶然性合格痕迹不同，行动者单元突触的合格痕迹也取决于行动者单元本身的活动。我们称之为偶然资格跟踪因为它取决于突触后活动。每一个突触上的合格痕迹都在不断地衰减，但随着突触前神经元的活动以及突触后神经元的激活而增加或减少。因子在 − (| 圣,)(15.3) 是积极的在 = 1 和消极。行动者单位资格痕迹中的突触后偶然性是批评家和行动者学习规则之间唯一的区别。通过保持信息采取了什么行动在什么州, 或有资格的痕迹让信贷奖励 (积极), 或归咎于惩罚 (负) 之间分配的政策参数 (药效的演员单位的突触) 根据这些参数的贡献, 单元的输出, 可能影响了后来的 值。或有资格的痕迹突触标记为他们应该如何修改改变单位的未来应对支持积极的 值。对于皮质纹状体突触的效率如何变化，批评家和演员学习规则提出了什么建议? 这两个学习规则都与 Donald Hebb 的经典提议有关，即只要突触前信号参与激活突触后神经元，突触的效率就会提高 (Hebb, 1949)。评论家和演员学习规则与赫伯的建议一致，即突触效能的变化取决于几个因素的相互作用。评论家学习规则的强化信号之间的互动是 和资格痕迹, 只取决于突触前的信号。神经学家称这为双因素学习规则，因为这是两个信号之间的相互作用

数量。演员学习规则, 另一方面, 是一个三因子学习规则, 因为除了取决于 , 其资格依赖于突触前和突触后活动痕迹。然而，与 Hebb 的建议不同的是，这些因素的相对时序对于突触效率的改变至关重要，因为合格的迹象会介入，使得强化信号能够影响最近活跃的突触。对于演员和评论家学习规则的信号时间的一些微妙之处值得关注。在定义神经元样的行为单元和批评单元时，我们忽略了需要突触输入来影响一个真正的神经元发射的时间。当突触前神经元的动作电位到达突触时，神经递质分子被释放出来，扩散到突触后神经元，在突触后神经元表面与受体结合; 这激活了导致突触后神经元兴奋的分子机制 (或者在抑制突触输入的情况下抑制其兴奋)。这个过程可能需要几十毫秒。但是，根据 (15.1) 和 (15.2)，输入到一个批评家和演员单元的同时会产生单元的输出。在 Hebbian-style 可塑性的抽象模型中，忽略这样的激活时间是很常见的，在这种模型中，突触效率会随着突触前和突触后活动的简单产物而变化。更现实的模型必须考虑激活时间。激活时间对于一个更现实的行动者单位来说尤其重要，因为它影响了偶然资格痕迹如何发挥作用，以便恰当地分配信贷以加强到适当的突触。表达式

在 − (| 圣,)? x(St) 根据上面给出的参与者学习规则，定义或有资格的跟踪包括后突触因素。

在 − (| 圣,)? 和突触前因子 x(St) 这是因为通过忽略激活时间，突触前活动 x(St) 参与导致突触后活动出现

在 − (| 圣,)?。指定信用担保- 正确地说，定义合格跟踪的突触前因子必须是也定义跟踪的突触后因子的原因。一个更现实的行为人单位的或有资格追踪必须考虑激活时间。(激活时间不应与神经元接收受该神经元活动影响的增强信号所需的时间混淆。资格跟踪的功能是跨越这个时间间隔，通常比激活时间长得多。我们将在下一节中进一步讨论这个问题。神经科学暗示了这个过程在大脑中的作用。神经科学家已经发现了一种 Hebbian 的可塑性，这种可塑性被称为“钉时依赖可塑性”(spike-timing-dependent plas 可塑性，STDP)，它为大脑中类似于动作体的突触可塑性的存在提供了可能。STDP 是一种 hebbian 式的可塑性，但突触效能的变化取决于突触前和突触后动作电位的相对时间。这种依赖性可以有不同的形式，但在研究最多的一种形式中，如果突触的尖峰在突触后神经元放电之前不久到达，突触的强度就会增加。如果时序关系逆转，突触后神经元触发后不久就会出现突触前突，那么突触的强度就会降低。STDP 是一种 Hebbian 的可塑性，它考虑了神经元的激活时间，是类动作学习所需要的要素之一。

STDP 的发现使神经科学家研究了 STDP 的三因素形式的可能性，即神经调节输入必须遵循适当的时间突触前和突触后峰值。这种形式的突触可塑性，称为奖赏调制 STDP，很像这里讨论的参与者学习规则。正常 STDP 产生的突触改变，只有在突触前突刺紧跟着突触后

的时间窗内有神经调节输入时才会发生。越来越多的证据表明，奖赏调制的 STDP 发生在背部纹状体的中等刺状神经元的棘突上，多巴胺提供神经调节因子——参与者学习发生在假定的行为批评算法的神经执行中，如图 15.5b 所示。实验已经证明了奖赏调节 STDP，只有当神经调节脉冲到达一个时间窗时，突触前突尖峰紧接突触后的 10 秒钟内，皮质纹状体突触的效率才会发生持久的变化 (Yagishita et al. 2014)。虽然这些证据是间接的，但这些实验指出，存在的偶然资格痕迹有长期的过程。产生这些痕迹的分子机制，以及可能低于 STDP 的更短的痕迹，还没有被理解，但是专注于时间依赖性和神经调节依赖性突触可塑性的研究仍在继续。我们在这里描述的神经类行为单元，以及它的效益型学习规则，以某种更简单的形式出现在 Barto 等人的演员-评论家网络中 (1983)。这个网络的灵感来源于生理学家 a • h • 克洛普夫 (1972,1982) 提出的“享乐神经元”假说。Klopf 的假设并不是所有的细节都与我们已经了解到的突触可塑性相一致，但是 STDP 的发现和不断增加的 STDP 奖赏调节形式的证据表明 Klopf 的想法可能并没有太大的偏离。接下来我们讨论 Klopf 的享乐神经元假说。

14.10 享乐神经元

在他的享乐主义神经元假说中，Klopf(1972, 1982) 推测，单个神经元试图最大限度地区分被视为奖励的突触输入和被视为惩罚的突触输入，它们通过调整自身行动潜力的奖励或惩罚结果的突触效率。换句话说，个体的神经元可以被训练成反应时序强化，就像动物可以被训练成工具调节任务一样。他的假设包括奖励和惩罚是通过同样的突触输入传递给神经元的，而突触输入会刺激或抑制神经元的刺产生活动。(如果 Klopf 知道我们今天对神经调节系统的了解，他可能会将增强作用赋予神经调节输入，但他希望避免任何集中的训练信息来源。) 过去突触前和突触后活动的突触局部痕迹在 Klopf 的假设中具有关键作用，即通过后来的奖励或惩罚来修改突触。他推测这些痕迹是由每个突触的分子机制实现的，因此不同于突触前神经元和突触后神经元的电活动。在

在这一章的参考文献和历史评论部分，我们要注意到其他人提出的一些类似的建议。Klopf 具体地推测，突触的功效是通过以下方式改变的。当一个神经元触发动作电位时，所有活跃于动作电位的突触都有资格接受它们的效率变化。如果动作电位在适当的时间内随着奖励的增加而增加，那么所有符合条件的突触的效率都会增加。对称地说，如果行为潜力在适当的时间内随着惩罚的增加而增加，符合条件的突触的效率就会降低。这是通过在突触前和突触后活动 (或者更确切地说，突触前活动和突触后活动的配对 (突触前活动参与导致的突触后活动) 上触发合格跟踪来实现的——我们称之为偶然合格跟踪。这本质上是前一节中描述的参与者单元的三因素学习规则。资格的形状和时间进程跟踪 Klopf 的理论反映了许多反馈循环的持续时间的神经元是嵌入式, 其中一些谎言完全在有机体的大脑和身体, 而其他延伸通过机体的外部环境由其运动和感觉系统。他的想法是，突触的资格追踪的形状就像一个直方图，是神经元被嵌入的反馈回路的持续时间。然后，合格跟踪的峰值将出现在该神经元参与的最普遍的反馈回路的持续时间。算法使用的资格痕迹在这本书里描述的简化版 Klopf 最初的想法, 是指数 (或几何) 减少功能控制的参数 和 。这简化了模拟和理论，但我们将这些简单的合格跟踪作为更接近 Klopf 原始概念的跟踪的占位符，通过改进信贷分配过程，在复杂的增强学习系统中具有计算优势。Klopf 的享乐主义神经元假说并不像一开始看起来那样难以置信。一个研究得很好的例子是一个寻求刺激而不寻求刺激的单一细胞是大肠杆菌。这种单细胞生物的运动受到环境中化学刺激的影响，即趋化行为。它在液体环境中游动，通过旋转附着在其表面的毛发状结构鞭毛。(是的, 它旋转时他们!) 细菌环境中的分子与细菌表面的受体结合。结合事件调节细菌逆转鞭毛旋转的频率。每一次倒转都会导致细菌在原地打滚，然后朝着一个随机的新方向前进。少量的化学记忆和计算导致当细菌游向其生存所需的高浓度分子 (引诱剂) 时鞭毛反转的频率降低，而当细菌游向有害分子 (驱虫剂) 的高浓度时，鞭毛反转的频率增加。其结果是，细菌倾向于坚持游上引诱剂梯度，并倾向于避免游上驱虫剂梯度。刚才描述的趋化行为叫做趋化作用。这是一种反复试验的行为，尽管不太可能涉及到学习: 细菌需

要少量的短期记忆来检测分子浓度梯度，但它可能不会维持长期记忆。人工智能先锋奥利弗

塞尔弗里奇将这种策略称为“运行和旋转”，并指出它作为一种基本的适应性策略的效用: “如果事情越来越好，继续以同样的方式前进，否则就四处移动”(塞尔弗里奇，1978,1984)。类似地，人们可能会认为神经元“游动”(当然不是字面意义上的“游动”) 是一种由它所嵌入的复杂反馈回路集合组成的媒介，其作用是获得一种输入信号，并避免其他信号。然而，与细菌不同的是，神经元的突触强度保留了有关其过去反复试验行为的信息。如果这个视图的一个神经元的行为 (或只是一种类型的神经元) 似乎是合理的, 那么如何神经元的闭环特性与环境互动是很重要的, 对于理解其行为, 神经元的环境由其他的动物与环境的动物作为一个整体进行交互。Klopf 的快乐主义神经元假说超越了单个神经元是增强学习因子的观点。他认为，智能行为的许多方面都可以被理解为一群自私自利的享乐主义神经元的集体行为的结果，这些神经元在构成动物神经系统的庞大社会或经济系统中相互作用。无论这种神经系统的观点是否有用，强化学习因子的集体行为对神经科学都有意义。我们接下来讨论这个问题。

14.11 集体强化学习

强化学习因子群体的行为与社会和经济系统的研究密切相关，如果克劳夫 (Klopf) 的享乐主义神经元假说是正确的，那么神经科学也是如此。上述假设 actor-critic 算法如何可能实现在大脑中只勉强地址这一事实的影响背侧和腹侧纹状体的细分, 演员和评论家的各自位置根据假设, 每个包含数以百万计的媒介棘神经元的突触接受改变调制相位的多巴胺神经元的活动。图 15.5a 中的 actor 是 k actor 单元的单层网络。该网络产生的行为是向量 (A1, A2，•••, Ak)? 假定驱动动物的行为。功效的突触的变化取决于所有这些单位强化信号 。因为演员单位试图使 尽可能大, 有效地为他们作为奖励的信号 (在这种情况下钢筋是一样的奖励)。因此，每个行动者单位本身就是一个强化学习主体——如果你愿意的话，它是一个享乐神经元。现在, 情况尽可能简单, 假设每一个单位接收相同的奖赏信号在同一时间 (尽管如此, 如上所示, 假设在所有层次的多巴胺释放突触在相同条件下和在同一时间可能是一个简化)。强化学习理论能告诉我们什么，当强化学习主体的所有成员根据一个共同的奖励信号学习时会发生什么? 多智能体强化学习领域考虑了多智能体群体学习的许多方面。虽然这个领域超出了这本书的范围，但我们相信它的一些基本概念和结果与思考有关

关于大脑的扩散神经调节系统。在多智能体强化学习 (和博弈论) 中，所有的智能体都试图最大化同时接收到的共同奖励信号的情形被称为合作博弈或团队问题。使团队问题变得有趣和具有挑战性的是，发送给每个代理的共同奖励信号评估整个人群产生的活动模式，也就是说，它评估团队成员的集体行动。这意味着任何单独的行为人对奖励信号的影响都是有限的，因为任何单独的行为人只对共同奖励信号所评估的集体行为的一个组成部分做出贡献。在这种情况下，有效的学习需要解决一个结构性的信用分配问题: 哪些团队成员，或团队成员，应该因为一个有利的奖励信号而得到赞扬，还是因为一个不利的奖励信号而受到指责? 这是一个合作的游戏，或者一个团队的问题，因为代理人在寻求增加相同的奖励信号: 在代理之间没有利益冲突。如果不同的代理接收到不同的奖励信号，那么这个场景将是一场竞争游戏，每个奖励信号再次评估人群的集体行动，每个代理的目标是增加自己的奖励信号。在这种情况下，代理之间可能存在利益冲突，这意味着对某些代理有利的行为对其他代理不利。甚至决定什么是最好的集体行动应该是一个不平凡的方面的博弈论。这种竞争环境可能也与神经科学有关 (例如，为了解释多巴胺神经元活动的异质性)，但这里我们只关注合作或团队的情况。一个团队中的每个强化学习代理如何学会“做正确的事情”，从而使团队的集体行动得到高度的奖励? 一个有趣的结果是, 如果每个代理可以有效学习尽管奖励信号被大量的噪音, 损坏, 尽管缺乏完整的状态信息, 那么人口作为一个整体将学会产生集体行动, 提高评估的常见奖励信号, 即使代理不能相互沟通。每个代理都面临着它自己的强化学习任务，它对奖励信号的影响被其他代理的影响深深地埋入了噪声中。事实上，对于任何代理，所有其他代理都是其环境的一部分，因为它的输入，传递状态信息的部分和奖励部分，都取决于其他代理的行

为。此外，由于无法访问其他代理的操作，甚至无法访问决定其策略的参数，每个代理只能部分地观察其环境的状态。这使得每个团队成员的学习任务非常困难, 但是如果每个使用强化学习算法能提高奖励信号即使在这些困难的条件下, 强化学习代理可以学会团队产生集体行动, 改善随着时间的推移评价团队的共同奖励的信号。如果团队成员是神经元样的单位，那么每个单位都必须有目标，随着时间的推移增加奖励的数量，就像我们在第 15.8 节中描述的，行动者单位所做的那样。每个单元的学习算法必须有两个基本特征。首先，它必须使用或有资格的追溯。回想一下，在神经术语中，偶然的资格跟踪是在突触前输入时启动 (或增加) 的

参与导致突触后神经元放电。与此相反，一个非偶然性的合格线索，是由突触前输入独立于突触后神经元的作用而启动或增加的。正如第 15.8 节所解释的那样，通过保存在哪些国家采取了什么行动的信息，或有资格的追溯，可以根据这些参数在决定代理人行动中所作的贡献的价值来分配给代理人的政策参数。通过类似的推理，团队成员必须记住它最近的行为，以便它可以根据随后收到的奖励信号增加或减少产生该行为的可能性。偶然资格跟踪的操作组件实现此操作内存。然而，由于学习任务的复杂性，或有资格仅仅是信贷分配过程中的一个初步步骤: 单个团队成员的行为与团队奖励信号的变化之间的关系是一个统计相关性，必须在许多试验中加以估计。或有资格是这一过程中必不可少的但初步的步骤。使用非偶然资格跟踪学习在团队设置中根本不起作用，因为它不提供一种将行为与随后的奖励信号变化联系起来的方法。非偶然资格跟踪对于学习预测是足够的，就像 actor - critics 算法的批评家部分所做的那样，但是它们不支持学习控制，就像 actor 组件必须做的那样。一个类似于 criticas 的群体的成员可能仍然会收到一个共同的强化信号，但是他们都将学会预测相同的数量 (在一个针对某一因素的方法中，这将是当前政策的预期回报)。每个人口成员在学习预测预期收益方面的成功程度将取决于它所获得的信息，而这对人口的不同成员可能是非常不同的。不需要人口产生不同的活动模式。这不是一个定义在这里的团队问题。团队问题中对集体学习的第二个要求是，团队成员的行为必须具有可变性，以便团队能够探索集体行动的空间。增强学习代理团队做到这一点的最简单方法是，让每个成员通过输出的持久性变化独立地探索自己的行为空间。这将导致整个团队改变其集体行动。例如，第 15.8 节中描述的一个参与者单元团队探索了集体行动的空间，因为每个单元的输出作为一个伯努利-逻辑单元，在概率上依赖于其输入向量分量的加权和。加权和偏差会使概率上升或下降，但总是存在可变性。由于每个单元都使用了一种增强策略梯度算法 (第 13 章)，所以每个单元在随机探索自己的行动空间的同时，调整自己的权重，目标是使自己所经历的平均奖励率达到最大。可以证明，正如 Williams(1992) 所做的那样，一个伯努利-逻辑强化单元的团队针对团队共同奖励信号的平均速率实现了一个整体的策略梯度算法，其中的行为是团队的集体行为。此外，Williams(1992) 表明，当团队中的各单元相互连接形成多层 ANN 时，使用增援的伯努利逻辑单元的团队会提升平均奖励梯度。在这种情况下，奖励信号被广播到所有单位

网络，尽管报酬可能只取决于网络输出单元的集体行动。这意味着伯努利-逻辑强化单元的多层团队学习起来就像被广泛使用的错误反向传播方法训练的多层网络，但在这种情况下，反向传播过程被广播的奖励信号所取代。在实践中，错误反向传播方法要快得多，但是强化学习团队方法作为一种神经机制更有可能，特别是考虑到在 15.8 节中讨论的关于奖赏调制 STDP 的知识。通过团队成员的独立探索进行探索是团队探索的最简单方式; 如果团队成员协调他们的行动，将注意力集中在集体行动空间的特定部分，或者通过相互通信，或者通过响应共同的输入，则可以使用更复杂的方法。还有一种机制比或有资格的机制更复杂，以解决结构性信贷分配问题，这在团队问题中更容易解决，因为可能的集体行动在某种程度上受到限制。一个极端的例子是一个赢家通吃的安排 (例如，大脑侧抑制的结果)，它限制了对只有一个或少数几个团队成员的集体行动。在这种情况下，获胜者会得到奖励或惩罚的奖励或责备。合作游戏 (或团队问题) 和非合作游戏问题的学习细节超出了这本书的范围。本章末尾的参考文献和历史评论部分引用了一些相关的出版物，包括对集体强化学习对神经科学的影响的广泛引用。

14.12 基于模型的大脑方法

强化学习在无模型和基于模型的算法之间的区别被证明对思考动物学习和决策过程是有用的。第 14.6 节讨论了这一区别如何与习惯性和目标导向的动物行为相一致。上述关于大脑如何实现一种行为-批评算法的假设与动物的习惯性行为模式有关，因为基本的行为-批评方法是无模型的。什么神经机制负责产生目标导向行为，它们如何与那些潜在的习惯性行为互动？研究与这些行为模式有关的大脑结构问题的一种方法是使老鼠大脑的某个区域失活，然后观察老鼠在结果减值实验中的行为 (第 14.6 节)。这些实验的结果表明，上面描述的演员-批评家假设在把演员放在背纹状体中过于简单。使背侧纹状体的一部分——背外侧纹状体 (DLS) 失去活性，削弱了习惯学习，使动物更依赖目标导向的过程。另一方面，使背侧纹状体 (DMS) 失活会削弱目标导向的过程，要求动物更多地依赖于习惯学习。像这样的结果支持了啮齿动物的 DLS 更多地涉及无模型的过程，而它们的 DMS 则更多地涉及基于模型的过程。的结果

在类似的实验中，用功能性神经成像对人类受试者进行的研究，以及对非人类灵长类的研究，都支持这样的观点，即灵长类大脑中的类似结构在习惯性和目标导向的行为模式中存在差异。其他研究发现，与基于模型的大脑前额叶皮层相关的活动，前额叶皮层的最前端部分与执行功能有关，包括计划和决策。具体涉及到的是眼窝前额皮质 (OFC)，前额皮质的部分位于眼睛上方。人类的功能神经成像，以及猴子单个神经元活动的记录，揭示了 OFC 中与生物意义刺激的主观奖励价值相关的强烈活动，以及与行动预期的奖励相关的活动。尽管这些结果并非毫无争议，但表明 OFC 在目标导向的选择中扮演了重要角色。这可能对动物环境模型的奖励部分至关重要。另一个涉及到基于模型的行为的结构是海马体，这是记忆和空间导航的关键结构。老鼠的海马体在老鼠以目标导向的方式在迷宫中穿行的能力中起着关键作用，这让托尔曼产生了动物在选择行为时使用模型或认知地图的想法 (第 14.5 节)。海马体也可能是人类想象新体验能力的重要组成部分 (Hassabis 和 Maguire, 2007;Barry Ólafsdóttir, 萨利姆, 哈萨比斯和施皮尔,2015)。最直接地暗示海马体参与规划的发现——在决策过程中需要一个环境模型——来自于解码海马体神经元活动的实验，以确定空间海马体活动的哪个部分在每时每刻的基础上表现出来。当老鼠在迷宫中的一个选择点停下来时，海马体中空间的表现会沿着动物可以从那个点出发的可能路径向前 (而不是向后)(Johnson and Redish, 2007)。此外，这些扫掠所代表的空间轨迹与老鼠随后的航行行为 (Pfeiffer and Foster, 2013) 密切相关。这些结果表明，海马体对于动物环境模型的状态转换部分至关重要，而海马体是一个系统的一部分，该系统使用模型来模拟未来可能的状态序列，以评估可能的行动过程的后果: 一种规划形式。上面描述的结果增加了大量关于目标导向或基于模型的学习和决策的神经机制的文献，但是许多问题仍然没有得到解答。例如，与 DLS 和 DMS 在结构上相似的区域如何能成为学习模式和行为模式的基本组成部分，这些模式和行为模式与无模型算法和基于模型的算法一样不同？独立的结构是否负责 (我们称之为) 环境模型的转换和奖励组件？是否所有的计划都是在决策时通过模拟未来可能的行动路线进行的，就像海马状突起的向前清扫活动所显示的那样？换句话说，是否都在计划类似于推出算法的东西 (第 8.10 节)？或者是模型有时在后台进行精炼或重新计算值信息，如 Dyna 体系结构 (第 8.2 节) 所示？大脑如何在习惯的使用和目标导向系统之间进行仲裁？事实上，神经之间是否有明显的分离

基质的系统？证据并没有指向最后一个问题的积极答案。多尔 (Doll)、西蒙 (Simon) 和 Daw(2012) 总结了这种情况，他们写道: “基于模型的影响在大脑处理奖励信息的地方或多或少都是普遍存在的。”这包括多巴胺信号本身，除了奖励预测错误被认为是无模型过程的基础之外，它还可以显示基于模型的信息的影响。通过强化学习的无模型和基于模型的区别，继续进行神经科学研究，有可能增强我们对大脑中习惯性和目标导向过程的理解。更好地理解这些神经机制，可能会导致将无模型方法和基于模型的方法结合在一起的算法，这种方法在计算强化学习中还没有被探索过。

14.13 上瘾

了解药物滥用的神经基础是神经科学的一个高度优先目标，有潜力为这一严重的公共健康问题开发新的治疗方法。一种观点认为，对药物的渴求是同样的动机和学习过程的结果，这些动机和学习过程使我们寻求满足我们生理需要的自然的有益体验。令人上瘾的物质，通过强烈的强化，有效地利用了我们的学习和决策的自然机制。这似乎是合理的，因为尽管不是所有的药物滥用都会直接或间接地增加多巴胺在纹状体神经元轴突终端区附近的水平，而纹状体的大脑结构与正常的基于奖励的学习密切相关 (第 15.7 节)。但是与毒瘾相关的自毁行为并不是正常学习的特征。多巴胺介导的学习有什么不同呢? 成瘾是正常学习的结果吗? 在我们进化的历史中，我们基本上无法获得这些物质，所以进化不能选择它们的破坏性影响。或者成瘾物质会以某种方式干扰正常的多巴胺介导的学习吗? 多巴胺神经元活动的奖赏预测错误假说及其与 TD 学习的联系是一个模型的基础，这个模型是由 Redish(2004) 提出的，但肯定不是成瘾的全部特征。该模型基于可卡因和其他成瘾药物的使用会产生短暂的多巴胺增加的观察。在模型中, 假设这种多巴胺增加增加 TD 错误, , 无法取消了价值函数的变化。换句话说, 而 是减少的程度, 一个正常的奖励是前期预测的事件 (15.6 节), 由于上瘾刺激的贡献并不减少随着奖励的信号变得预测: 不能“预测药物的回报。“模型通过阻止 成为负奖励信号时由于一种上瘾的药物, 从而消除 TD 学习状态的纠错功能与政府相关的药物。其结果是这些状态的值不受约束地增加，使导致这些状态的行动优先于所有其他状态。

与 Redish 的模型相比，上瘾行为要复杂得多，但是这个模型的主要思想可能是一个谜。或者这个模型可能具有误导性。多巴胺似乎在所有成瘾类型中都没有发挥关键作用，而且并不是每个人都同样容易产生成瘾行为。此外，该模型不包括伴随慢性药物服用而产生的许多回路和大脑区域的变化，例如，这些变化会导致药物在反复使用后效果逐渐减弱。成瘾还可能涉及基于模型的过程。尽管如此，Redish 的模型说明了如何利用强化学习理论来理解一个主要的健康问题。同样地，强化学习理论在新的计算精神病学领域的发展中也有影响，该领域旨在通过数学和计算方法提高对精神障碍的理解。

14.14 总结

大脑奖赏系统所涉及的神经通路是复杂而不完全了解的，但是神经科学的研究正朝着理解这些通路及其在行为中的作用迅速进展。这项研究揭示了大脑奖励系统和强化学习理论之间惊人的对应关系。多巴胺神经元活动的奖赏预测错误假说是由科学家提出的，他们认识到 TD 错误的行为与产生多巴胺的神经元的活动有惊人的相似之处，多巴胺是哺乳动物奖励相关的学习和行为所必需的神经递质。实验在 1980 年代末和 1990 年代在实验室的神经学家 Wolfram 舒尔茨表明多巴胺神经元响应的事件与大量的活动, 称为相位的反应, 只有在动物并不认为这些事件, 这表明多巴胺神经元信号奖励预测错误, 而不是奖励本身。此外，这些实验表明，当动物学习根据先前的感觉线索来预测一个有意义的事件时，多巴胺神经元的阶段活动转变为早期的预测线索，而减少为后期的预测线索。这与 TD 误差作为增强学习剂学习预测奖励的效果相似。其他实验结果坚定地证明，多巴胺神经元的阶段性活动是学习的强化信号，通过多巴胺产生神经元的大量分支，到达大脑的多个区域。这些结果是一致的回报之间的区别我们信号,Rt, 强化信号,TD 的错误 t 我们目前的大多数算法。多巴胺神经元的阶段反应是强化信号，而不是奖励信号。一个突出的假设是，大脑执行某种类似于行为-批评算法的东西。大脑中的两种结构 (纹状体的背侧和腹侧分支) 在基于奖励的学习中都扮演着重要的角色，它们可能分别像演员和批评家一样发挥作用。TD 错误是行动者和批评家的强化信号，这与多巴胺神经元轴突瞄准纹状体的背侧和腹侧亚区这一事实吻合得很好; 多巴胺似乎是在两个结构中调节突触可塑性的关键; 对神经调节器的目标结构的影响，如多巴胺，取决于目标结构的性质，而不仅仅是神经调节器的性质。

演员和批评家可以由神经单元组成的 ANNs 来实现，这些神经单元具有基于第 13.5 节描

述的政策梯度演员-批评家方法的学习规则。这些网络中的每一个连接就像大脑中神经元之间的突触，学习规则对应于控制突触效率如何随着突触前神经元和突触后神经元活动的功能而变化的规则，以及与多巴胺神经元输入相对应的神经调节输入。在此设置中，每个突触都有自己的合格跟踪记录涉及该突触的过去活动。演员和评论家学习规则之间唯一的区别是, 他们使用不同的资格的痕迹: 评论家单元的痕迹 non-contingent 因为他们不涉及评论家单元的输出, 而演员单位的痕迹或有, 因为除了演员单位的输入, 他们依靠演员单元的输出。假设在大脑中实施一个行为-批评系统，这些学习规则分别对应于控制皮质纹状体突触可塑性的规则，这些突触将信号从皮质传递到背侧和腹侧纹状体的主要神经元，这些突触也接受来自多巴胺神经元的输入。

一个演员单元的学习规则在演员-评论家网络中，与奖赏调节的 spiker -时间依赖的可塑性密切相关。在 spike-timing 依赖性可塑性 (STDP) 中，突触前和突触后活动的相对时间决定了突触改变的方向。在奖赏调制的 STDP 中，突触的变化除了依赖于神经调节器 (如多巴胺) 外，还依赖于在满足 STDP 条件后能持续 10 秒的时间窗。越来越多的证据表明，奖赏调节的 STDP 发生在皮质纹状体突触，在这里，行动者的学习发生在假定的行为-批评系统的神经实施中，这增加了假设的合理性，即某些动物的大脑中存在着类似行为-批评系统的东西。

突触合格性的概念和行动者学习规则的基本特征来源于 Klopf 的“享乐神经元”假说 (Klopf, 1972, 1981)。他推测，单个的神经元寻求获得奖励，并通过调整其突触的功效，以奖励或惩罚其动作电位的结果来避免惩罚。神经元的活动会影响其后期的输入，因为神经元内嵌在许多反馈回路中，其中一些在动物的神经系统和身体内，而另一些则通过动物的外部环境。Klopf 认为，如果突触参与了神经元的激活 (使其成为资格跟踪的偶然形式)，那么它们就暂时被标记为有资格进行修改。如果增强信号在突触合格时到达，则对突触的有效性进行修改。我们提到了一个细菌的趋化行为，它是一个单细胞的例子，它指导它的运动，以寻找一些分子并避开其他分子。

多巴胺系统的一个显著特征是，纤维向大脑的多个部分广泛释放多巴胺。虽然很可能只有一些多巴胺神经元群会发出同样的强化信号，但如果这个信号到达参与肌动类型学习的许多神经元的突触，那么情况就可以了

被建模为一个团队问题。在这种类型的问题中，一个增强学习代理集合中的每个代理接收相同的增强信号，该信号依赖于集合中的所有成员或团队的活动。如果每个团队成员都使用了一个足够强大的学习算法，那么即使团队成员之间没有直接的交流，团队也可以通过全局广播增强信号来提高整个团队的性能。这与多巴胺信号在大脑中的广泛分散是一致的，并且为训练多层网络提供了一种神经上可行的替代方法。无模型强化学习和基于模型的强化学习之间的区别有助于神经科学家研究习惯性学习和目标导向学习和决策的神经基础。到目前为止的研究表明，他们的大脑中有一些区域比其他区域更参与一种类型的过程，但是目前的情况还不清楚，因为没有模型的和基于模型的过程在大脑中似乎并没有整齐地分开。许多问题仍未得到解答。也许最有趣的是，海马体，一个传统上与空间导航和记忆有关的结构，似乎参与了模拟未来可能的行动路线，作为动物决策过程的一部分。这表明，它是使用环境模型进行规划的系统的一部分。强化学习理论也影响着对药物滥用背后神经过程的思考。基于奖励预测误差假设，建立了药物成瘾特征的模型。它提出，像可卡因这样的成瘾刺激物会使 TD 学习产生与毒品摄入有关的行为价值的无限增长。这还远不是一个完整的成瘾模型，但它说明了如何从计算的角度提出理论，可以通过进一步的研究来检验。计算精神病学的新领域同样侧重于计算模型的使用，其中一些模型来自于强化学习，以更好地理解精神障碍。本章仅触及强化学习的神经科学与计算机科学与工程强化学习的发展如何相互影响的表面。增强学习算法的大多数特性都是纯粹基于计算的考虑，但也有一些特性受到神经学习机制的假设的影响。值得注意的是，随着实验数据的积累，大脑的奖励过程，许多纯计算动机的增强学习算法的特点，都是与神经科学数据一致的。计算强化学习的其他特性, 如资格痕迹和团队的能力强化学习代理学习集体行动的影响下一个 globally-broadcast 强化信号, 也可能变成平行实验数据作为神经科学家继续解开半球动物的学习和行为的神经基础。

14.15 书目的和历史的言论

在这本书中，关于学习和决策的神经科学与强化学习的方法之间的相似性的出版物数量是巨大的。我们只能举出一小部分。《新和合》(2009)、《大安》(2008)、《金彻》(2011)、《勒德维格》、《贝勒玛》、《皮尔森》(2011)、《沙阿》(2012) 都是不错的起点。强化学习理论与经济学、进化生物学和数学心理学一起，正在帮助建立人类和非人类灵长类的神经机制选择的定量模型。这一章以学习为中心，只略微涉及决策的神经科学。Glimcher(2003) 引入了“神经经济学”领域，强化学习有助于从经济学的角度研究决策的神经基础。参见《一瞥》和《费尔》(2013)。Dayan 和 Abbott(2001) 关于神经科学的计算和数学建模的文本包括强化学习在这些方法中的作用。Sterling 和 Laughlin(2015) 研究了学习的神经基础，研究了能够有效适应行为的通用设计原则。

15.1 基础神经科学有许多很好的论述。坎德尔, 施瓦兹、Jessell Siegelbaum 和 Hudspeth(2013) 是一个权威和非常全面的来源。

15.2 Berridge 和 Kringelbach(2008) 回顾了奖励和快乐的神经基础，指出奖励处理有许多维度和涉及许多神经系统。空间阻止了对 Berridge 和 Robinson(1998) 有影响的研究的讨论，他们区分了刺激的享乐影响，他们称之为“喜欢”，以及他们称之为“渴望”的动机效应。Hare O ’Doherty、Camerer、Schultz 和 Rangel(2008) 从经济学的角度研究了价值相关信号的神经基础，区分了目标值、决策值和预测错误。决策价值是目标价值减去行动成本。参见兰格尔, 卡默勒, 蒙塔古 (2008), 兰赫尔和兔 (2010), 彼得斯和布鲁里溃疡˙ 秋儿 (2010)。15.3 多巴胺神经元活动的奖赏预测误差假说是最多的舒尔茨、大安和蒙塔古 (1997) 在显著位置上进行了讨论。这个假设是由蒙塔古、达扬和塞杰诺斯基 (1996) 首次明确提出的。正如他们所说的假设，它指的是奖励预测误差 (RPEs)，但不是特别针对 TD 误差; 然而，他们对假设的发展清楚地表明，他们指的是 TD 错误。最早认识到的 TD-error/多巴胺连接，我们知道的是 Montague, Dayan, Nowlan, Pouget 和 Sejnowski(1993)，他们提出了一种由 Schultz 组的多巴胺信号的结果驱动的，由 TD-error 调制的 Hebbian 学习规则。Quartz、Dayan、Montague 和 Sejnowski(1992) 在一篇摘要中也指出了这种联系。Montague 和 Sejnowski(1994) 强调了预测在大脑中的重要性，并概述了如何通过弥漫性神经系统 (如多巴胺系统) 实现由 TD 错误调制的 Hebbian 学习预测。托诺尼,Friston Reeke、斯波恩 Edelman(1994) 提出了一个大脑中价值依赖性学习的模型，在这个模型中，突触的变化是由全球神经调节信号 (虽然他们没有单独列出多巴胺) 提供的 td 样错误所介导的。mont -tague, Dayan, Person 和 Sejnowski(1995) 提出了一个利用 TD 误差的蜜蜂觅食模型。该模型基于 Hammer, Menzel 和同事的研究 (Hammer and Menzel, 1995; 哈默 (1997) 显示神经调节物质章鱼胺在蜜蜂中起强化信号的作用。Montague 等人 (1995) 指出，多巴胺可能在脊椎动物的大脑中起着类似的作用。Barto (1995a) 将 actor -批评家架构与 basal-ganglionic 电路联系起来，并讨论了 TD 学习与 Schultz 团队的主要结果之间的关系。胡克、亚当斯 (Adams) 和巴托 (Barto)(1995) 建议道明学习和表演评论家架构如何映射到基底神经节的解剖学、生理学和分子机制。Doya 和 Sejnowski(1998) 扩展了他们关于鸟鸣学习模型的早期论文 (Doya 和 Sejnowski,1995)，包括一个与多巴胺识别的 td 样错误，以加强选择要记忆的听觉输入。O ’Reilly 和 Frank(2006) 和 O ’Reilly、Frank、Hazy 和 Watz(2007) 认为，阶段性多巴胺信号是 RPEs 而不是 TD 错误。为了支持他们的理论，他们引用了变量间刺激间隔的结果，这些结果与简单的 TD 模型的预测不匹配，以及观察到二阶条件作用之外的高阶条件作用很少被观察到，而 TD 学习也不是那么有限。Dayan 和 Niv(2008) 讨论了强化学习理论和奖励预测错误假说如何与实验数据相一致的“善、恶、丑”。格里姆彻 (2011) 回顾了支持奖励预测误差假说的实证结果，强调了该假说对当代神经科学的重要性。

15.4 Graybiel(2000) 是一篇关于基底神经节的简介。提到的实验涉及 optogenetic 多巴胺神经元的激活是由蔡, 张 Adamantidis, 存根,Bonci,de Lecea 和戴瑟罗斯 (2009),Steinberg Keiflin,Boivin, 威滕, 戴瑟罗斯, 和 Janak(2013), 和 Claridge-Chang Roorda,Vrontou,Sjulson, 李, 赫希,Miesenböck(2009)。Fiorillo、Yun 和 Song(2013)、Lammel、Lim 和 Malenka(2014)、Saddoris、Cacciapaglia、Wightmman 和 Carelli(2015) 等研究表明，多巴胺神经元的信号特

性是针对不同目标区域的。rpe 信号神经元可能属于多种多巴胺神经元中的一种，它们具有不同的靶点和不同的功能。Eshel, Tian, Bukwich，和 Uchida(2016) 发现在小鼠的经典条件作用下，在侧 VTA 中多巴胺神经元的奖赏预测错误反应的同质性，尽管他们的结果并不排除在更广泛的区域内的反应多样性。Gershman、Pesaran 和 Daw(2009) 研究了强化学习任务，这些任务可以用单独的奖励信号分解成独立的部分，并在人类神经成像数据中找到证据，表明大脑利用了这种结构。

15.5 舒尔茨 1998 年的调查文章是一个很好的之间´e 到非常广泛的文学关于奖励预测多巴胺神经元的信号。Berns, McClure, Pagnoni 和 Montague (2001)，Breiter, Aharon, Kahneman, Dale, and Shizgal (2001)，Pagnoni, Zink, Montague 和 Berns (2002)，O 'Doherty, Dayan, Friston, Critchley，和 Dolan(2003) 描述了脑功能成像研究支持在人脑中存在像 TD 错误这样的信号。

15.6 本节大致按照 Barto(1995 年 a) 的说法解释 TD 错误是如何模拟的舒尔茨小组对多巴胺神经元的阶段反应的主要结果。

15.7 这部分主要是基于高桥、舍恩鲍姆、和合本 (2008) 和《圣经》(2009)。据我们所知，Barto(1995 年 a)、Houk、Adams 和 Barto(1995 年) 首先推测了在基底神经节中可能实现的行为批判算法。O 'Doherty、Dayan、Schultz、Deichmann、Friston 和 Dolan(2004) 在对人体进行仪器调节时进行功能性磁共振成像的基础上，提出行动者和批评家最可能分别位于背侧和腹侧纹状体。Gershman、Moustafa 和 Ludvig(2014) 关注了时间在基底神经节强化学习模型中的表现方式，讨论了各种计算方法对时间表示法的证据和含义。本节所描述的演员-评论家架构的假设神经实现中几乎没有关于已知的基底神经节解剖学和生理学的细节。除了 Houk、Adams 和 Barto(1995) 的更为详细的假设之外，还有一些其他的假设包括与解剖学和生理学更具体的联系，并声称可以解释更多的数据。这些假设包括 Suri 和 Schultz (1998, 1999)，Brown, Bullock 和 Grossberg (1999)，Contreras-Vidal 和 Schultz (1999)，Suri, Bargas, Arbib (2001)，O' reilly and Frank (2006)，O' reilly, Frank, Hazy，和 Watz(2007)。乔尔 (Joel, Niv) 和鲁平 (Ruppin)(2002) 批判性地评估了其中几个模型的解剖学合理性，并提出了一种替代方案，以适应一些被忽视的基底神经节回路的特征。

15.8 此处讨论的演员学习规则比 in 中的规则要复杂得多 Barto 等人早期的演员-评论家网络 (1983)。Actor-unit 资格痕迹在网络的痕迹就在 ××(St) 而不是完整的 (− (| 圣,))x(St)。这项工作没有受益于第 13 章提出的政策梯度理论或威廉姆斯的贡献 (1986 年，1992 年)，他展示了伯努利-逻辑单元的 ANN 是如何实现政策梯度方法的。Reynolds 和 Wickens(2002) 提出了皮质纹状体通路中突触可塑性的三因素规则，其中多巴胺调节皮质纹状体突触效能的变化。他们讨论了这种学习规则的实验支持和可能的分子基础。的示范 spike-timing-dependent 可塑性 (STDP) 是归因于马克拉姆, 陆¨bke,Frotscher, 和 Sakmann(1997), 从早期的实验证据 Levy 和 Steward(1983) 等人认为突触前和突触后峰值的相对时间对于诱发突触效率的改变至关重要。Rao 和 Sejnowski(2001) 提出 STDP 是如何由 TD-like 机制产生的，这种机制存在于具有非偶然资格的突触上，持续时间约为 10 毫秒。Dayan(2002) 认为这需要一个错误，就像 Sutton 和 Barto (1981a) 经典条件作用的早期模型，而不是一个真正的 TD 错误。大量关于奖赏调制 STDP 的文献中有代表性的出版物有 Wickens (1990)，Reynolds 和 Wickens (2002)，calab 西方雷西，Picconi, Tozzi 和 Di Filippo(2007)。Pawlak 和 Kerr(2008) 的研究表明，多巴胺对于诱导 STDP 是必要的。参见 Pawlak, Wickens, Kirkwood 和 Kerr(2010)。Yagishita、Hayashi-Takagi、Ellis-Davies、Urakubo、Ishii 和 Kasai(2014) 发现，多巴胺只在 STDP 刺激后 0.3 秒至 2 秒的时间窗内促进小鼠中等刺状神经元的脊柱增大。Izhikevich(2007) 提出并探讨了使用 STDP 时序条件触发或有资格跟踪的想法。Frémaux Sprekeler, 郭士纳 (2010) 提出的理论条件成功的学习规则基于 reward-modulated STDP。

15.9 Klopf 的享乐主义神经元假说 (Klopf 1972, 1982) 启发了我们的演员-批评家算法实现为一个具有单个神经元样单元的 ANN，称为 actor 单元，实现一个类似于效率的学习规则 (Barto, Sutton, and Anderson, 1983)。与 Klopf 的突触-局部资格相关的想法已经被其他人提出。Crow(1968) 提出，皮质神经元突触的变化对神经活动的后果是敏感的。他强调需要解决

神经活动的时间延迟和它对突触可塑性的奖赏调节形式的影响，他提出了一种偶然的资格形式，但与整个神经元相关，而不是单个的突触。根据他的假设，一波神经活动导致在波中所涉及的细胞的短期变化，这些细胞是从没有被激活的细胞的背景中挑选出来的。这样的细胞是由于对奖励信号的短期改变而变得敏感……以这样一种方式，如果这样的信号在变化的衰减时间结束之前发生，细胞间的突触连接就会变得更加有效。(乌鸦,1968) 乌鸦反对先前的提议, 反射神经回路扮演这一角色, 指出奖励的影响信号电路将“…建立突触联系导致混响 (也就是说, 那些参与活动时的奖励信号), 而不是那些路径导致自适应运动输出。Crow 进一步假设奖励信号是通过一个“独特的神经纤维系统”传递的，这可能是 Olds 和 Milner(1954 年) 所使用的神经纤维系统，它可以将突触连接“从短期转变为长期形式”。在另一种有远见的假设中，米勒提出了一种类似于效果的学习规则，其中包括了突触局部的或有资格的追踪:

…据设想，在特定的感官环境下，神经 B 碰巧会触发“有意义的活动爆发”，然后转化为运动行为，然后改变情况。必须假定有意义的在神经层面上，突发对它自己的所有突触都有影响在那个时候很活跃……因此，初步选择要加强的突触，虽然还没有真正加强它们。…加强信号……做最后的选择…和完成确定的变化在适当的突触。(米勒,1981,p . 81)

米勒的假说还包括 critic-like 机制, 他称之为“感官分析器装置,”根据经典条件作用工作原理为神经元提供强化信号, 这样他们会学习从低股价较高的国家, 从而预测 TD 的使用错误的强化信号 actor-critic 架构。米勒的想法不仅与 Klopf 的想法相似 (除了明显地调用了一个独特的“增强信号”之外)，还预测了奖赏调制 STDP 的一般特性。Seung(2003) 称之为“享乐主义突触”(hedonistic synapse) 的一个相关但不同的观点是，突触以效应定律的方式分别调整了释放神经递质的可能性: 如果释放后有奖励，释放的可能性就会增加，如果释放失败后奖励会减少。这与明斯基 1954 年在普林斯顿大学博士论文中使用的学习计划本质上是一样的，他把这种类似突触的学习元素称为 SNARC(随机神经模拟强化计算器)。偶然资格也与这些想法有关，尽管它取决于单个突触的活动而不是突触后神经元。也有关于 Unnikrishnan 和 Venugopal(1994) 的提议，该提议使用了 Harth 和 Tzanakou(1974) 的基于相关的方法来调整 ANN weights。Frey 和 Morris(1997) 提出了“突触标记”的概念，用于诱导突触效能的长期增强。尽管与 Klopf 的资格相似，他们的标签假设是由一个突触的暂时强化组成的，这个突触可以通过随后的神经元激活转化为一个长期的强化。O 'Reilly and Frank(2006) 和 O 'Reilly (Frank)、Frank、Hazy 和 Watz(2007) 的模型使用工作记忆来桥接时间间隔而不是资格痕迹。Wickens 和 Kotter(1995) 讨论了突触资格的可能机制。He, Huertas, Hong, Tie, Hell, Shouval, Kirkwood(2015) 提供了证据，证明了时序过程中皮层神经元突触中是否存在偶然性合格的痕迹，就像那些合格的痕迹 Klopf 假设的那样。Barto(1989) 讨论了使用与细菌趋化有关的学习规则的神经元的隐喻。Koshland 对细菌趋化性的广泛研究的部分原因是细菌的特征和神经元特征的相似性 (Koshland, 1980)。参见伯格 (1975)。Shimansky(2009) 提出了一个突触学习规则有点类似于上面提到的，每个突触都像一个趋化细菌。在这种情况下，一组突触“游动”到突触权重值的高维空间吸引物。Montague, Dayan, Person 和 Sejnowski(1995) 提出了一种蜜蜂觅食行为的化学模拟模型，涉及神经调质章鱼胺。

15.10 加强学习主体在团队和游戏中的行为研究问题有很长的历史，大致分为三个阶段。据我们所知，第一阶段是由俄国数学家和物理学家 m.l. Tsetlin 进行的。他的一组作品在 1966 年去世后被出版为《采编林》(1973)。我们的第 1.7 和 4.8 节涉及到他关于学习自动机的研究。Tsetlin 收藏还包括学习自动机的研究团队和游戏问题, 导致以后的工作在这一领域使用随机学习自动机作为被纳兰德拉和 Thathachar(1974、1989),Viswanathan 和 Narendra(1974),Lakshmivarahan 和 Narendra(1982), 纳兰德拉和惠勒 (1983), 和 Thathachar Sastry(2002)。Thathachar 和 Sastry(2011) 是一个更全面的描述。这些研究大多局限于非联想学习自动机，这意味着它们没有涉及到联想或语境的强盗问题 (2.9 节)。第二阶段从学习自动机的扩展到联想，或情境，案例开始。Barto、Sutton、Brouwer(1981) 和 Barto 和 Sutton(1981) 在单层 ANNs 中实验了联合随机学习自动机，并向其广播了一个全局增强信号。学习算法是 Harth 和 Tzanakou(1974) 的 Alopex 算法的联合扩展。Barto 等人将这种学习关联搜索元素 (ase) 称为神经元类元素。Barto 和阿南丹 (1985) 介绍了一个关联强化学习

算法称为关联 reward-penalty(AR−P) 算法。将随机学习自动机理论与模式分类理论相结合，证明了算法的收敛性。Barto(1985、1986) 和 Barto 和约旦 (1987) 描述的结果与基于“增大化现实”技术团队 −P 单元连接成多层人工神经网络, 显示他们可以学习非线性函数,XOR 等,globally-broadcast 强化信号。Barto(1985) 广泛地讨论了 ANNs 的这种方法，以及这种学习规则是如何与当时文献中的其他人相关的。Williams(1992) 对这门学习规则进行了数学分析和拓展，并将其应用于训练多层神经网络的误差反向传播方法。Williams(1988) 描述了将反向传播和强化学习结合起来进行 ANNs 训练的几种方法。威廉姆斯 (1992) 表明,AR−P 算法的一个特例是一个增强的算法, 虽然得到了更好的结果与普通 AR−P 算法 (Barto,1985)。

增强学习剂团队的第三个兴趣阶段受到了对多巴胺作为一种广为传播的神经调节器的作用的理解的增强和对奖赏调节 STDP 存在的推测的影响。与早期的研究相比，这项研究更关注突触可塑性的细节和神经科学的其他约束。出版物包括以下 (按时间顺序和按字母顺序): 巴特利特和巴克斯特 (1999、2000), 谢和 Seung(2004), 巴拉和梅尔 (2007),Farries 和 Fairhall(2007), 御马 (2007),Izhikevich(2007),Pecevski, 马斯河, 和 Legenstein(2008),Legenstein,Pecevski, 和马斯河 (2008),Kolodziejski,Porr, 和我们¨rgö 水獭 (2009),Urbanczik 森 (2009), 和 Vasilaki Frémaux,Urbanczik, 森, 郭士纳 (2009)。NowéVrancx,De Hauwere(2012) 回顾了最近的多智能体强化学习在更广阔的领域发展

Yin 和 Knowlton(2006) 回顾了货币贬值专家的研究成果对啮齿类动物的研究表明，习惯性行为和目标导向行为 (心理学家使用这个短语) 分别与背外侧纹状体 (DLS) 和背侧纹状体 (DMS) 的处理有关。Valentin, Dickinson 和 O ’Doherty(2007) 在《结果-贬值》中对人类受试者的功能成像实验结果表明，眶额皮层 (OFC) 是目标导向选择的重要组成部分。Padoa-Schioppa 和 Assad(2006) 的单单元录音支持了 OFC 在编码价值指导选择行为方面的作用。Rangel, Camerer，和 Montague(2008) 和 Rangel and Hare(2010) 从神经经济学的角度回顾了大脑如何做出目标导向的决策。Pezzulo、van der Meer、Lansink 和 Pennartz(2014) 回顾了内部生成序列的神经科学，并提出了这些机制如何可能是基于模型规划的组成部分的模型。Daw 和 Shohamy(2008) 提出，虽然多巴胺信号与习惯性的或无模型的行为有很好的联系，但是其他的过程涉及目标导向的，或基于模型的行为。由 Bromberg-Martin、Matsumoto、Hong 和 Hikosaka(2010) 所做的实验数据表明，多巴胺信号包含与习惯和目标导向行为相关的信息。多尔 (Doll)、西蒙 (Simon) 和多瓦 (Daw)(2012) 认为，在大脑中，可能没有明确的划分机制来服务于习惯性学习和目标导向的学习和选择。

15.12 Keiflin 和 Janak(2015) 回顾了 TD 错误和成瘾之间的联系。Nutt, Lingford-Hughes, Erritzoe 和 Stokes(2015) 批判性地评估了成瘾是由多巴胺系统紊乱引起的假设。Montague、Dolan、Friston 和 Dayan(2012) 概述了计算精神病学领域的目标和早期努力，Adams、Huys 和 Roiser(2015) 回顾了更近期的进展。

15. 第十六章应用和案例研究

在这一章中，我们提出了一些强化学习的案例研究。其中一些是具有潜在经济意义的重要应用。其中之一，塞缪尔的西洋跳棋手，主要是历史兴趣。我们的演示旨在说明在实际应用程序中出现的一些权衡和问题。例如，我们强调领域知识是如何融入到问题的制定和解决中去的。我们还强调了对成功的应用程序通常至关重要的表示问题。在这些案例研究中使用的算法要比我们在本书其余部分中介绍的算法复杂得多。强化学习的应用还远未达到常规，通常需要和科学一样多的艺术。使应用程序更简单、更直接是当前强化学习研究的目标之一。

15.1 TD-Gammon

迄今为止最令人印象深刻的强化学习应用之一是由杰拉德·特劳罗 (Gerald Tesauro) 在《backgammon 游戏》(Tesauro, 1992, 1994, 1995, 2002) 的游戏。Tesauro 的计划 TD-Gammon 只需要很少的西洋双陆棋知识，但却能很好地发挥，接近世界上最强的大师水平。TD-Gammon 的学习算法的一个简单的组合 TD( ) 算法和非线性函数近似使用多层人工神经网络 (ANN) 的训练,backpropagating TD 错误。西洋双陆棋是一种主要的游戏，在世界各地都有，有许多比赛和常规的世界锦标赛比赛。这在一定程度上是一场碰运气的游戏，而且它是一种流行的工具，因为它能筹集大量的资金。也许有更多的职业西洋双陆棋棋手比有职业棋手。这款游戏由 24 个地点组成的棋盘上的 15 个白棋和 15 个黑棋组成。在下一页的右边显示了游戏早期的一个典型位置，从白人玩家的角度来看。怀特刚刚掷出骰子，得到 5 和 2。这意味着他可以移动他的作品 5 步 1 步

一片白色逆时针方向移动 12 3 3 4 5 6 7 8 9 10 11 12 18 17 16 14 13 19 19 20 21 22 23 24 黑块移动顺时针

西洋双陆棋的位置 (可能是同一件)2 个步骤。例如，他可以从 12 点移动两段，一段移动到 17 点，另一段移动到 14 点。怀特的目标是把他所有的棋子推进最后一个象限 (19-24 分)，然后离开棋盘。第一个把所有棋子都拿走的玩家获胜。一个复杂的问题是，这些碎片相互作用，因为它们在不同的方向上彼此擦肩而过。例如，如果它是布莱克的移动，他可以用骰子滚 2 把一块从 24 点移动到 22 点，“击中”白色部分。被击中的棋子被放置在棋盘中间的“条”上 (我们已经在那里看到了一个先前被击中的黑色棋子)，在那里他们从一开始就重新进入了比赛。但是，如果一个点上有两个棋子，那么对手就不能移动到那个点; 这些碎片不受撞击。因此，白棋不能使用他的 5-2 掷骰在 1 点上移动他的任何一个棋子，因为它们可能产生的点数被一组黑色棋子占据。形成连续的被占据点块来阻挡对手是游戏的基本策略之一。《西洋

双陆棋》涉及到更多的并发症，但是上面的描述提供了基本的思想。有了 30 个零件和 24 个可能的位置 (26 个，包括杆和板)，应该可以清楚地看到，可能的西洋双陆棋位置的数量是巨大的，远远超过任何物理可实现的计算机的内存元素的数量。从每个位置移动的可能性也很大。对于一个典型的骰子滚动，可能有 20 种不同的游戏方式。考虑到未来的动作，比如对手的反应，你必须考虑到可能的掷骰子。结果是，博弈树的有效分支因子约为 400。这实在是太大了，以至于不能有效地使用传统的启发式搜索方法，这种方法在象棋和跳棋等游戏中是如此有效。另一方面，这个游戏与 TD 学习方法的能力是很好的匹配。虽然游戏是高度随机的，但是游戏状态的完整描述在任何时候都是可用的。游戏通过一系列的移动和位置进行演变，最终以一方或另一方的胜利告终，结束游戏。这个结果可以被解释为一个最终的回报。另一方面，我们迄今所描述的理论结果不能有效地应用于这项任务。状态的数量如此之多，以至于无法使用查找表，而对手是不确定性和时间变化的来源。TD-Gammon 使用一种非线性的 TD( )。估计价值,v̂(s,w), 任何国家 (板位置) 是为了赢的概率估计从状态。为了达到这个目标, 奖励被定义为零的时间所有步骤除游戏赢了。为了实现 value 函数，TD-Gammon 使用一个标准的多层 ANN，就像下一页右边显示的那样。(真正的

Vt + 1 Vt 隐藏的单位

西洋双陆棋位置 (198 输入单元) 预测获胜的可能性

TD 错误, ·····圣 + 1 v̂(w)−v̂(St,w) TD 的错误 v̂(St,w) 隐藏的单位 (40 - 80) 图 16.1:TD-Gammon ANN 网络在其最后一层有两个额外的单元来估计每个玩家以一种特殊的方式获胜的概率，这种方式被称为“gammon”或“backgammon”。该网络的输入是一个 backgammon 位置的表示，输出是该位置的值的估计值。在第一个版本的 TD-Gammon, TD-Gammon 0.0 中，西洋双陆棋的位置以一种相对直接的方式呈现在网络上，这种方式涉及到很少的西洋双陆棋知识。那样, 但是，要对人工神经网络的工作原理和信息的最佳呈现方式有深入的了解。注意 Tesauro 所选择的精确表示是很有意义的。网络总共有 198 个输入单元。对于西洋双陆棋棋盘上的每一个点，四个单位表示在这个点上的白色棋子的数量。如果没有白色部分，那么所有四个单位的值都是 0。如果有一件，那么第一个单位就值为 1。这编码了“污点”的基本概念，即。一种可以被对手击中的棋子。如果有两个或多个部件，则第二个单元设置为 1。这就编码了一个基本概念，即对手不能在其上着陆。如果这个点上正好有 3 个单位，那么第三个单位就设为 1。这就编码了“单一备用”的基本概念，即。除了这两件作品之外，还有另外一件作品。最后，如果有三个以上的部件，那么第四个部件将被设置成与超过三个部件的数量成比例的值。让 n 表示点金币的总数, 如果 n > 3, 那么第四单元的值 (n−3)/2。它在给定的点上编码了“多个备件”的线性表示。在这 24 个点上，白人 4 个单位，黑人 4 个单位，总共 192 个单位。两个额外的单位编码黑白块在酒吧的数量 (每个值 n / 2, 其中 n 是碎片的数量栏), 和两个编码黑白作品的数量已经成功地从棋盘上拿掉 (这些值 n / 15, 其中 n 是碎片的数量已经承担了)。最后，两个单位以二进制的方式表示，无论是白色的还是黑色的。这些选择背后的一般逻辑应该是清楚的。基本上，Tesauro 试图以一种简单的方式表示位置，同时保持单位的数量相对较少。他为每一个可能相关的概念上不同的可能性提供了一个单位，他将它们的范围大致扩大到相同的范围，在这个例子中是 0 到 1 之间。给定一个西洋双陆棋位置的表示，网络以标准的方式计算其估计值。对应于从输入单元到隐藏单元的每个连接都是实值权重。每个输入单元的信号被乘以它们相应的权重，在隐藏的单位上求和。隐藏单元 j 的输出 h(j) 是加权和的一个非线性的 s - id 函数:

习近平在第 i 的值是输入单元和维琪 j 隐藏连接单元的重量 (所有网络传感器组合在一起形成了参数的权重向量 w)。乙状结肠的输出总是在 0 和 1 之间, 并有一个自然的解释是基于概率求和的证据。从隐藏单元到输出单元的计算完全类似。从隐藏单元到输出单元的每个连接都有一个单独的权重。输出单元形成加权和，然后通过相同的 s 形非线性进行传递。TD-Gammon TD 的 semi-gradient 形式使用 ( )12.2 节中描述的算法, 与梯度计算的误差反向传播算法 (Rumel-hart、辛顿和威廉姆斯,1986)。回想一下，这种情况的一般更新规则是

wt + 1 . = wt +

Rt + 1 + v̂(wt) 圣 + 1−v̂(St,wt)

zt 型, (16.1)

其中 wt 是所有可修改参数的矢量 (在本例中是网络的权重)，而 zt 是一个合格跟踪的矢量，每个 wt 的每个组件都有一个，由?

zt 型。= zt−1 + v̂(St,wt), 与 z0 。= 0。这个方程的梯度可以用反相法有效地计算出来 gation 过程。西洋双陆棋应用程序中, = 1 和奖励总是零除了获胜,TD 的错误部分的学习规则通常只是 v̂(w) 圣 + 1−v̂(St,w), 是显示在图 16.1。为了应用学习规则，我们需要一个西洋双陆棋游戏的来源。通过在比赛中学习西洋双足球员，他获得了连续不断的比赛。为了选择它的动作，TD-Gammon 考虑了大约 20 种掷骰子的方式以及相应的位置。产生的位置是第 6.8 节所讨论的后态。该网络被用来评估他们的价值。然后选择将导致估计值最高的位置的移动。继续以这种方式，随着 TD-Gammon 对双方的移动，很容易产生大量的西洋双陆棋游戏。每个游戏都被视为一个插曲，位置序列作为状态，S0, S1, S2，…Tesauro 使用非线性 TD 规则 (16.1) 完全增量地，也就是说，在每个单独的移动之后。网络的权值最初设定为小的随机值。因此，最初的评价完全是武断的。因为这些动作是在这些评价的基础上进行的，所以最初的动作是不可避免的，而最初的游戏通常会在一方或另一方获胜之前，持续数百或数千步，几乎是偶然的。然而，在几十场比赛之后，成绩迅速提高。在与自己打了大约 30 万场比赛之后，TD-Gammon 0.0 就像上面描述的那样，学会了和之前最好的西洋双陆棋一样玩

项目。这是一个惊人的结果，因为所有先前的高性能计算机程序都使用了大量的西洋双陆棋知识。举个例子，当时的冠军项目，可以说是“神经 gammon”，另一个由 Tesauro 编写的程序，使用的是 ANN，而不是 TD 学习。Neurogammon 的网络被训练在一个由西洋双陆棋专家提供的大型训练语料库上，此外，开始时还为西洋双陆棋特别设计了一套特征。Neurogammon 是一个高调频的，高效的 backgammon 项目，在 1989 年决定性地赢得了世界 backgammon 奥运会。另一方面，TD-Gammon 0.0 的构造基本为零。它能做的和神经 gammon 一样，所有其他的方法都证明了自我游戏学习方法的潜力。TD- gammon 0.0 的锦标赛成功与零专家的 backgammon 知识建议一个明显的修改: 增加专门的 backgammon 特性，但保留自玩 TD 学习方法。这就造成了 TD-Gammon 1.0。TD-Gammon 1.0 显然比以前所有的 backgammon 程序都要好，并且只在人类专家中发现了严重的竞争。后来版本的程序，TD-Gammon 2.0(40 个隐藏单元) 和 TD-Gammon 2.1(80 个隐藏单元)，通过一个选择性的两层搜索程序进行扩充。要选择动作，这些程序不仅要看马上会出现的位置，还要看对手可能掷出的骰子和动作。假设对手总是采取对他最有利的动作，每个候选动作的预期值都被计算出来，最好的结果被选中。为了节省计算机时间，第二次搜索只对在第一次搜索之后排名较高的候选动作进行，平均大约有四五个动作。双层搜索只影响选定的移动; 学习过程和以前一样。最终版本的 TD-Gammon 3.0 和 3.1 使用了 160 个隐藏单元和选择性的三层搜索。TD-Gammon 举例说明学习值函数和决策时间搜索在启发式搜索和 MCTS 方法中的结合。在后续的工作中,Tesauro 和加尔佩林 (1997) 探讨了轨迹采样方法代替宽屏搜索, 减少错误率的生活玩大数值因素 (4 x-6x) 同时保持 5 - 10 秒的思考时间合理的举动。在 20 世纪 90 年代，特索罗能够在许多比赛中与世界级的人类选手比赛。结果摘要载于表 16.1。

程序隐藏单位 40 80 40 80 80 80 80 80 80 培训游戏 30 万 80 万 150 万对手结果 TD-Gammon 0.0 TD-Gammon 1.0 TD-Gammon 2.0 TD-Gammon 2.1 TD-Gammon 3.0 其他节目罗伯特，马格里尔，……各种大师罗伯特 • 卡扎罗斯的节目并列最佳 ——13 分/ 51 游戏 7 分/ 38 游戏 −1 pt / 40 游戏 + 6 分/ 20 场比赛

表 16.1:TD-Gammon 结果摘要

基于这些结果和西洋双陆棋大师的分析 (Robertie, 1992;TD-Gammon 3.0 似乎在接近或可能比世界上最优秀的人类选手发挥的力量要好。Tesauro 在随后的一篇文章 (Tesauro, 2002) 中报道了对 TD-Gammon 相对于顶级玩家的移动决策和加倍决策进行了广泛的分析的结果。得出的结论是，TD-Gammon 3.1 在计件运动决策中具有“不平衡优势”，在加倍决策中具有“微弱优势”，超过了人类。TD-Gammon 对最优秀的人类玩家的游戏方式产生了重大影响。例如，它学会了在某些开局位置上的表现与人类最好球员之间的惯例不同。基于 TD-Gammon 的成功和进一步的分析，现在最优秀的人类玩家可以像 TD-Gammon 一样扮演

这些角色 (Tesauro, 1995)。当其他几个受 TD-Gammon 启发的 ANN backgammon 项目，如水母、Snowie 和 GNUBackgammon 被广泛使用时，对人类游戏的影响大大加快。这些项目使 ANNs 产生的新知识得以广泛传播，从而大大提高了人类锦标赛的整体水平 (Tesauro, 2002)。

15.2 塞缪尔跳棋的球员

泰索罗的 TD-Gammon 的一个重要先驱是阿瑟 • 萨缪尔 (1959 年，1967 年) 在构建跳棋程序方面的开创性工作。塞缪尔是第一个有效利用启发式搜索方法的人，我们现在称之为时间性差异学习。他的跳棋棋手除了具有历史意义外，还是很有启发性的个案研究。我们强调撒母耳方法与现代强化学习方法的关系，并试图传达撒母耳使用这些方法的动机。

1952 年，塞缪尔首次为 IBM 701 编写了一个下棋程序。他的第一个学习项目在 1955 年完成，并在 1956 年在电视上展示。后来版本的程序获得了良好的，但不是专家，演奏技巧。塞缪尔被游戏作为机器学习的一个研究领域所吸引，因为游戏比“从生活中拿走”的问题要简单，同时还允许对启发式过程和学习如何一起使用进行富有成果的研究。他选择学习跳棋而不是象棋，因为它的相对简单性使人们有可能更注重学习。

塞缪尔的程序通过在每个当前位置执行一个前瞻搜索来发挥作用。他们使用我们现在所说的启发式搜索方法来确定如何扩展搜索树以及何时停止搜索。每个搜索的终端位置都被一个值函数或“得分多项式”用线性函数逼近来评价或“得分”。在这方面和其他方面，塞缪尔的作品似乎受到了香农 (1950) 的建议的启发。特别是，塞缪尔的计划是基于香农的极小化过程，以找到最佳的移动从目前的位置。在搜索树中从得分终端位置向后移动，每一个位置都得到最佳移动的位置的分数，假设机器总是试图最大化分数，而对手总是试图最大化分数

最小化。塞缪尔称这个位置为“支撑得分”。当极小极大过程到达搜索树的根——当前位置——它产生了最佳的移动，假设对手将使用相同的评价标准，转移到它的观点。一些版本的 Samuel 的程序使用复杂的搜索控制方法，类似于所谓的“alpha-beta” cutoffs(例如，参见 Pearl, 1984)。塞缪尔采用了两种主要的学习方法，最简单的一种叫做死记硬背。它只包括保存游戏中遇到的每个棋盘位置的描述，以及由极大极小化过程确定的支撑值。结果是，如果已经遇到的位置再次作为搜索树的终端位置出现，那么搜索的深度就会被有效地放大，因为这个位置的存储值缓存了之前进行的一个或多个搜索的结果。最初的一个问题是，该项目没有被鼓励沿着最直接的途径取得胜利。萨缪尔给了它一种“方向感”，即在极小极大分析期间，当一个位置每次被备份到一个水平 (称为层) 时，它的价值就会减少一小部分。“如果程序现在面临的选择是董事会的职位，其分数只与层数不同，它将自动做出最有利的选择，如果赢了，选择低层的选择; 如果输了，选择高层的选择”(Samuel, 1959，第 80 页)。塞缪尔发现这种类似折扣的技巧对于成功的学习至关重要。死记硬背的学习产生了缓慢但持续的改进，这对于开始和结束游戏是最有效的。他的程序在从许多游戏中学习到对抗自己、各种各样的人类对手，以及在监督学习模式下的书本游戏之后，成为了一个“比一般人更好的新手”。

死记硬背和塞缪尔作品的其他方面强烈地暗示了时间差异学习的基本思想——一个国家的价值应该等于可能遵循的国家的价值。Samuel 在他的第二种学习方法中最接近这个想法，他的“泛化学习”程序修改了值函数的参数。塞缪尔的方法在概念上和后来泰索罗在《泰德-加蒙》中使用的方法是一样的。他在自己的程序中与另一个版本的程序进行了很多游戏，每次移动后都进行更新。Samuel 更新的想法是由图 16.2 中的备份图建议的。每个开环表示程序下一步移动的位置，一个正在移动的位置，每个实环表示对手下一步移动的位置。对每一方移动后的每个移动位置的值进行更新，从而产生第二个移动位置。这次更新是针对从第二个移动位置启动的搜索的极小值。因此，总体效果是对真实事件的一个完整移动进行备份，然后对可能的事件进行搜索，如图 16.2 所示。由于计算的原因，塞缪尔的实际算法要比这个复杂得多，但这是基本的思想。塞缪尔没有明确的奖励。相反, 他固定的重量最重要的功能, 这篇文章的优势特性, 测量块的数量项目相对于其对手有多少, 给国王, 更高的权重, 包括改进, 以便更好的贸易当胜利比失败。因此，塞缪尔计划的目标是提高他的计分优势，这在跳棋中

与获胜高度相关。

图 16.2: 萨缪尔跳棋运动员的备份图。

然而，塞缪尔的学习方法可能漏掉了一个声音时间差算法的一个重要部分。时间性差异学习可以看作是一种使价值函数与自身一致的方法，这一点我们可以从塞缪尔的方法中清楚地看到。但还需要一种将值函数与状态的真实值绑定的方法。我们通过奖励和贴现或给终端状态一个固定的值来实现这一点。但萨缪尔的方法没有奖励，也没有对游戏的最终位置进行特殊处理。正如塞缪尔自己所指出的那样，他的价值函数仅仅通过给所有的位置赋予一个恒定的值就可以变得一致。他希望通过给他的计件优势条款一个巨大的、不可改变的权重来阻止这种解决方案。但是，尽管这可能会降低找到无用的评估函数的可能性，但它并没有禁止它们。例如，一个常数函数仍然可以通过设置可修改的权值来获得，从而抵消不可修改权值的影响。

因为塞缪尔的学习过程并不局限于寻找有用的评价函数，所以它本应该随着经验而变得更糟。事实上，萨缪尔报告说他在大量的自我游戏训练中观察到了这一点。为了让项目再次得到改善，塞缪尔不得不进行干预，将绝对值最大的权重设为零。他的解释是，这种激烈的干预使程序脱离了局部最优状态，但另一种可能性是，它使程序脱离了评价函数，这些函数是一致的，但与游戏的成败无关。

尽管有这些潜在的问题，萨缪尔的西洋跳棋玩家使用泛化学习方法接近“好于平均水平”的游戏。相当不错的业余对手将其描述为“狡猾但可击败”(塞缪尔，1959)。与死记硬背的版本相比，这个版本能够开发一个好的中间游戏，但是在开始和结束的游戏中仍然很弱。这个程序还包含了一种搜索一系列特性的能力，以找到在形成 value 函数时最有用的特性。后来的版本 (Samuel, 1967) 在搜索过程中加入了一些改进，比如 alpha-beta 剪枝，广泛使用被称为“书本学习”的监督学习模式，以及称为“签名表”(Griffith, 1966) 的分层查找表，以表示值函数而不是线性函数近似。这个版本比 1959 年的版本学得好多了，虽然还没有达到大师的水平。塞缪尔的跳棋程序被广泛认为是人工智能和机器学习方面的重大成就。

15.3 沃森的双倍下注

IBM Watson1 是一个由 IBM 研究人员开发的系统，用于播放广受欢迎的电视智力竞赛节目《危险边缘》。2011 年，它在与人类冠军的表演赛中获得一等奖，一举成名。尽管沃森所展示的主要技术成就是它能够快速准确地回答自然语言的问题，而不是广泛的知识领域，但它的胜利是危险的! 游戏的关键部分还依赖于复杂的决策策略。Tesauro, Gondek, Lechner, Fan 和 Prager(2012, 2013) 采用了上面描述的 Tesauro 的 TD-Gammon 系统，创造了 Watson 在《每日加倍》(Daily-Double) (DD) 中所使用的策略，在与人类冠军的比赛中赢得胜利。这些作者报告说，这种下注策略的有效性远远超出了人类玩家在实时游戏中所能做的，而且它与其他高级策略一起，是沃森令人印象深刻的获胜表现的重要因素。在这里，我们只关注 DD 的赌注，因为沃森的组件更多地依赖于强化学习。

冒险! 由三名选手扮演，他们面对着一个显示 30 个方块的棋盘，每个方块都隐藏着一个线索，并且有一美元的价值。正方形被分成六列，每一列对应一个不同的类别。参赛者选择一个方块，主持人阅读方块的提示，每个参赛者可以选择按蜂鸣器 (“嗡嗡”) 回应提示。第一个参与比赛的选手试图对这个线索做出回应。如果这个参赛者的回答是正确的，那么他们的分数将以正方形的美元值增加; 如果他们的回答不正确，或者他们在五秒内没有回答，他们的分数就会下降这么多，而其他参赛者就有机会对同样的线索做出回应。一个或两个方块 (取决于游戏当前的回合) 是特殊的 DD 方块。选择其中一个的选手得到一个独家的机会来回应这个方的线索，并且必须在线索被揭示之前做出决定——赌多少钱，赌多少。赌注必须大于 5 美元，但不能超过选手的当前分数。如果参赛者对 DD 线索反应正确，他们的分数将增加赌注金额; 否则，它会因下注金额而减少。每一场比赛的结尾都是一场“最后的危险”(FJ) 游戏，每位参赛者都要写下一个密封的赌注，然后在阅读线索后写下一个答案。三轮比赛后

得分最高的选手 (每一轮包含 30 条线索) 获胜。游戏还有很多其他的细节，但是这些已经足够让人欣赏了

1 IBM 公司注册商标。2 .《危险产品》的注册商标。

DD 赌博的重要性。输赢往往取决于参赛者的 DD 赌博策略。当沃森选择 DD 广场, 它选择了赌值, 通过比较行动问ˆ(年代, 赌), 估计赢的概率从当前游戏状态, 为每个 round-dollar 年代, 法律选择。除了下面描述的一些降低风险的措施，Watson 选择了具有最大动作值的赌注。当需要进行博彩决策时，通过使用两种估计来计算动作值，这两种估计是在任何现场游戏发生之前就已经学会的。第一个是选择每个合法赌注后各州的估计值 (第 6.8 节)。这些估计从州值函数, 获得 v̂(• w), 定义为参数 w, 让赢的概率的估计沃森从任何游戏状态。第二个用于计算动作值的估计值给出了“类内 DD 置信度”pDD，该估计值估计了 Watson 对尚未发现的 DD 线索做出正确响应的可能性。

Tesauro 等人使用的强化学习方法学习上述 TD-Gammon v̂(• w): 一个简单的非线性组合 TD( ) 使用多层安重量 w 培训 backpropagating TD 错误在很多模拟游戏。状态由专门为 Jeopardy 设计的特征向量表示到网络上。特征包括三名玩家当前的分数、剩余的 DDs 数量、剩余线索的总价以及与游戏剩余游戏数量相关的其他信息。与 TD-Gammon self-play 学到, 沃森的 v̂ 就是数以百万计的模拟比赛精明狡猾的人类玩家的模型。在类别内的信心估计取决于沃森在当前类别中给出的正确答案 r 和错误答案 w 的数量。对 (r, w) 的依赖关系是根据沃森对数千个历史类别的实际准确性来估计的。以前学到的价值函数 v̂, 集中 DD 信心 pDD, 沃森计算 q̂(年代, 打赌) 为每个法律 round-dollar 选择如下:

问ˆ(年代, 打赌)= pDD×v̂(SW + 打赌。)+(1−pDD)×v̂(SW− 打赌。)(16.2), 西南是沃森的当前的分数, 和 v̂ 给游戏状态的估计价值沃森的 DD 反应线索后, 这是正确的或不正确的。计算这样一个行动值对应的洞察力从 3.19 运动动作值给定的期望下一个状态值操作 (除了在这儿下 afterstate 预期值, 因为整个游戏的下一个状态取决于下一个平方的选择)。

Tesauro 等人发现，通过最大化动作值来选择赌注会产生“令人恐惧的风险”，这意味着如果 Watson 对线索的反应恰好是错误的，那么这个损失对于它获胜的机会来说可能是灾难性的。为了降低错误答案的下行风险，Tesauro 等人通过减去 Watson 正确/错误的后态评估的一小部分标准差来调整 (16.2)。他们进一步降低了风险，因为他们禁止了将导致错误回答的事后价值降低到一定限度以下的赌注。这些措施略微降低了沃森对获胜的预期，但它们显著降低了下跌风险，不仅是在平均风险的风险下，而且在极端风险的情况下，风险中性的沃森会押注大部分或全部的资金。

为什么 TD-Gammon self-play 不习惯学习临界值的方法函数 v̂? 在危险中自娱自乐! 因为沃森和其他任何人类选手都不一样，所以不会有很好的表现。自我游戏将导致探索国家空间区域，这不是对抗人类对手，特别是人类冠军的典型方式。此外，不像西洋双陆棋，危险! 这是一个信息不完全的游戏，因为参赛者不能接触到所有影响他们对手比赛的信息。特别是, 冒险! 选手们不知道他们的对手对各种各样的线索有多大的信心。自玩就像和一个和你持相同牌的人玩扑克一样。

由于这些复杂性，开发沃森的 DD-wagering 策略的大部分努力都被用于创建人类对手的良好模型。这些模型没有涉及到游戏的自然语言方面，而是游戏中可能发生的事件的随机过程模型。统计数据是从一个广泛的粉丝创建的存档游戏信息从展览开始到现在。档案包括信息，如排序的线索，正确和错误的参赛者答案，DD 地点，DD 和 FJ 的赌注，为近 30 万线索。构建了三个模型: 一个平均参赛模型 (基于所有数据)，一个冠军模型 (基于 100 个最佳玩家的游戏统计)，一个大冠军模型 (基于 10 个最佳玩家的游戏统计)。除了在学习过程中充当对手外，这些模型还被用来评估学习的 DD-wagering 策略所带来的好处。沃森在模拟中使用基线启发式的 DD-wagering 策略时的胜率为 61%; 当它使用学习值和一个默认的置信值时，它的赢球率增加到 64%; 而对生活类别的置信度，则是 67%。Tesauro 等人认为这是一个显著的改进，因为 DD 在每场比赛中只需要 1.5 到 2 次。

因为沃森只有几秒钟的时间来打赌，以及选择方块并决定是否加入，所以做出这些决定所需的计算时间是一个关键因素。的安实现 v̂ 允许 DD 的赌注是足够快以满足时间约束的生活。

然而, 一旦游戏可以通过改善仿真软件模拟的速度不够快, 在比赛快结束的时候估计是可行的投资的价值在许多蒙特卡罗试验平均每个选择的结果是由模拟发挥游戏的结束。选择基于蒙特卡罗试验而不是 ANN 的现场比赛中的 endgame DD 投注明显提高了 Watson 的表现，因为在终场比赛中价值估计的错误会严重影响 Watson 获胜的机会。通过蒙特卡罗试验做出所有的决定可能会带来更好的赌注决定，但考虑到游戏的复杂性和实时游戏的时间限制，这是完全不可能的。虽然它快速准确地回答自然语言问题的能力是沃森的主要成就，但它所有复杂的决策策略都导致了它令人印象深刻的击败人类冠军。根据 Tesauro 等 (2012):

…很明显，我们的策略算法实现了超过人类能力的定量精确和实时性能。

这一点在 DD 赌博和游戏结束时的嗡嗡声中尤为明显，在这种情况下，人类根本无法与沃森所做的精确的公平、信心估计以及复杂的决策计算相匹配。

15.4 优化内存控制

大多数计算机以动态随机存取存储器 (DRAM) 作为主要存储器，因为它的成本低、容量大。DRAM 存储器控制器的工作是有效地利用处理器芯片和非芯片 DRAM 系统之间的接口，提供高速程序执行所需的高带宽和低延迟数据传输。内存控制器需要处理动态变化的读/写请求模式，同时遵守硬件所需的大量时间和资源限制。这是一个可怕的调度问题，特别是对于具有多个内核共享相同 DRAM 的现代处理器来说。我˙ 油漆、Mutlu 集市´ınez,(2008) 和 Caruana(也集市´ınez 我˙ 油漆,2009) 设计了一种强化学习内存控制器和显示, 它可以大大提高程序执行的速度是可能的与传统控制器的时候他们的研究。它们的动机是现有最先进的控制器的限制，这些控制器使用的策略没有利用过去的调度经验，也没有考虑到调度决策的长期后果。我˙ 油漆等的项目是由模拟的手段, 但是他们设计控制器的硬件实现, 包括所需的详细级别学习 algorithm-directly 处理器芯片。访问 DRAM 涉及许多步骤，这些步骤必须根据严格的时间限制来完成。DRAM 系统由多个 DRAM 芯片组成，每个芯片都包含以行和列排列的多个矩形存储单元阵列。每个电池在电容器上储存一点电荷。由于电荷随时间减少，每个 DRAM 单元需要每隔几毫秒重新充电一次，以防止内存内容丢失。这需要刷新单元格，这就是为什么 DRAM 被称为“dynamic”。每个单元格数组都有一个行缓冲区，其中包含一行位元，这些位元可以转移到数组的某一行或其中的某一行。激活命令“打开一行”，这意味着将其地址由命令指示的行内容移动到行缓冲区。当一行打开时，控制器可以向单元格数组发出读写命令。每个 read 命令将行缓冲区中的一个单词 (连续位的短序列) 传输到外部数据总线，每个 write 命令将外部数据总线中的一个单词传输到行缓冲区。在打开不同的行之前，必须发出一个 precharge 命令，该命令将行缓冲区中的 (可能更新的) 数据传输回单元阵列的寻址行。在此之后，另一个 activate 命令可以打开要访问的新行。读和写命令是列命令，因为它们顺序地将位传递到行缓冲区的列或列之外; 可以在不重新打开行的情况下传输多个位。对当前打开的行执行读写命令比访问不同的行要快，这将涉及额外的行命令: 预充和激活; 这是有时

被称为“行局部性”。“内存控制器维护一个内存事务队列，该队列存储共享内存系统的处理器的内存访问请求。控制器必须通过向内存系统发出命令来处理请求，同时遵守大量的时间限制。一个控制器的调度访问请求的策略可以对内存系统的性能产生很大的影响，比如可以满足请求的平均延迟，以及系统能够实现的吞吐量。最简单的调度策略按照请求到达的顺序处理访问请求，方法是在开始为下一个请求提供服务之前发出请求所需的所有命令。但是，如果系统没有为这些命令之一做好准备，或者执行一个命令会导致资源被充分利用 (例如，由于服务一个命令而导致的时间限制)，那么在完成旧的请求之前就开始处理新请求是有意义的。策略可以通过重新排序请求来提高效率，例如，将读请求的优先级放在写请求之上，或者将读/写命令的优先级放在已经打开的行上。称为“先到先得”、“先到先得”(FR-FCFS) 的策略将列命令 (读和写) 置于行命令 (激活和预充) 之上，并且在连接时优先级为最老的命令。在通常遇到的情况下，FR-FCFS 在平均内存访问延迟方面表现优于其他调度策略 (Rixner, 2004)。图 16.3 是我的高级视图˙ 油漆等的强化学习内存控制器。他们将 DRAM 访问过程建

模为 MDP，其状态是事务队列的内容，其操作是对 DRAM 系统的命令: 预充电、激活、读取、写入和 NoOp。当动作被读或写时，奖励信号为 1，否则为 0。状态转换被认为是随机的，因为系统的下一个状态不仅依赖于调度程序的命令，而且还取决于调度程序无法控制的系统行为的各个方面，比如处理器内核访问 DRAM 系统的工作负载。

图 16.3: 增强学习 DRAM 控制器的高级视图。调度程序是强化学习代理。它的环境由事务队列的特性表示，它的操作是对 DRAM 系统的命令。c ? IEEE 2009。经许可转载, 从 j . f . 集市´ınez 和大肠我˙ 油漆, 动态多核资源管理: 机器学习的方法, 微,IEEE,29(5),p。12。

MDP 的关键是对每个状态中可用的操作的约束。回忆从第三章组可用的操作依赖于状态: (St), 在时间步的行动 t 和可用的行动 (St) 是一组在州圣在这个应用程序中,DRAM 的完整性保证系统不允许违反时间或资源约束的行为。虽然我˙ 油漆等人并不明确, 他们有效地完成这种根据所有可能状态的设置 (St)。这些约束解释了为什么 MDP 有一个 NoOp 操作，为什么奖励信号是 0，除了发出读或写命令。NoOp 是指一个州内唯一的法律行为。最大化利用的内存系统, 控制器的任务是驱动的系统状态可以选择读或写操作: 只有这些行动导致通过外部数据总线发送数据, 所以只有这些, 有助于系统的吞吐量。虽然 precharge 和 activate 不会立即产生回报，但是代理需要选择这些操作，以使以后选择奖励的读和写操作成为可能。

调度代理使用 Sarsa(第 6.4 节) 学习动作-值函数。国家由 6 个整数值的特征表示。为了逼近动值函数，算法采用线性函数近似，采用散列编码实现线性函数近似 (第 9.5.4 节)。瓷砖编码有 32 个倾斜，每一个都存储 256 个动作值作为 16 位定点数字。探索 -greedy = 0.05。状态特性包括事务的读请求队列的数量, 数量的事务中写请求队列, 写请求的数量的事务队列等待他们的行被打开, 和读请求的数量在事务队列等待他们行被打开, 他们发行的最古老的请求处理器。(其他特性取决于 DRAM 如何与缓存内存交互，这里省略了一些细节)。国家特征的选择是基于我˙ 油漆等的 DRAM 性能因素的影响的理解。例如，根据事务队列中每个事务队列的数量来平衡服务读和写的速率可以帮助避免延迟 DRAM 系统与缓存内存的交互。作者实际上生成了一个相对较长的潜在特性列表，然后使用逐步特性选择指导的模拟将它们缩减为几个。将调度问题作为 MDP 表示的一个有趣的方面是，用于定义动作值函数的块编码的特性输入与用于指定动作约束集 A(St) 的特性不同。虽然块编码输入来自事务队列的内容，但是约束集依赖于许多与时间和资源约束相关的其他特性，这些特性必须由整个系统的硬件实现来满足。这样，动作约束保证了学习算法的探索不会危及物理系统的完整性，而学习被有效地限制在硬件实现更大状态空间的“安全”区域。

因为这项工作的一个目标是，学习控制器可以在芯片上实现，以便在计算机运行时在线学习，硬件实现细节是重要的考虑因素。设计包括两个五阶段管道，用于计算和比较每个处理器时钟周期的两个动作值，以及更新适当的操作值。这包括访问存储在静态 RAM 中的芯片上的块代码。配置我˙ 油漆等人模拟, 这是一个 4 ghz 基于芯片的典型高端工作站时他们的研究, 有 10 个处理器周期为每个 DRAM 周期。考虑到填充管道所需的周期，在每个 DRAM 循环中最多可以评估 12 个操作。我˙ 油漆等人发现法律命令, 任何一个国家都是很少的数量大于, 性能损失是微不足道的如果足够的时间并不总是可以考虑所有合法的命令。这些巧妙的设计细节使得在多处理器芯片上实现完整的控制器和学习算法成为可能。

我˙ 油漆等人评价他们的学习控制器仿真通过比较它与其他三个控制器:1) 上述 FR-FCFS 控制器产生最好的平均性能,2) 常规控制器处理每个请求, 和 3) 理想无法实现的控制器, 控制器称为乐观, 能够维持 100% DRAM 吞吐量如果有足够的需求, 忽略了所有的时间和资源约束, 否则建模 DRAM 延迟 (如行缓冲撞击) 和带宽。他们模拟了由科学和数据挖掘应用程序组成的 9 个内存密集型并行工作负载。图 16.4 显示了 9 个应用程序的每个控制器的性能 (执行时间的倒数，归一化为 FR-FCFS 的性能)，以及它们在应用程序中的性能的几何平均值。在图中标注为 RL 的学习控制器比 FR-FCFS 提高了 7% 到 33%，平均提高了 19%。由于芯片实现的基本原理的学习算法是允许在线调度策略适应不断变化的工作负载, 我˙ 油漆等人分析了在线学习的影响比之前学的固定的政策。他们训练

图 16.4: 在一组 9 个模拟基准应用程序中，4 个控制器的性能。控制器是: 最简单的“有序”控制器、FR-FCFS、学习控制器 RL 和不可实现的乐观控制器，它们忽略了所有的时间

和资源约束提供一个性能上限。性能，归一化到 FR-FCFS 的性能，是倒数的执行时间。最右边是每个控制器在 9 个基准应用程序上性能的几何平均值。控制器 RL 最接近理想的性能。2009 c ? IEEE。经许可转载, 从 j . f . 集市´ınez 和大肠我¨ 油漆, 动态多核资源管理: 机器学习方法，Micro, IEEE, 29(5)，p. 13。

它们的控制器使用来自所有 9 个基准应用程序的数据，然后在整个应用程序的模拟执行过程中保持结果操作值不变。他们发现，在线学习的控制器的平均性能比使用固定策略学习的控制器的平均性能好 8%，这使他们得出结论，在线学习是他们方法的一个重要特性。这种学习记忆体控制器从来没有致力于物理硬件，因为制造成本很高。不过, 我¨ 油漆等人可以令人信服地说他们的仿真结果的基础上, 一个内存控制器, 通过强化学习网上学习有可能提高性能水平, 否则需要更复杂和更昂贵的记忆系统, 同时删除从人类设计师的一些负担需要手动设计高效的调度策略。Mukundan 和集市´ınez(2012) 把这个项目提前调查学习控制器与额外的动作, 其他性能标准, 使用遗传算法和更复杂的奖励函数派生。他们考虑了与能源效率有关的其他性能标准。这些研究的结果超过了前面描述的结果，并且在他们考虑的所有性能标准上显著地超过了 2012 年的最先进水平。这种方法对于开发复杂的 power-aware DRAM 接口特别有希望。

15.5 人机级的游戏

将增强学习应用于实际问题的最大挑战之一是决定如何表示和存储价值函数和/或策略。除非状态集是有限的，并且足够小，可以通过查找表来进行详尽的表示——就像我们的许多示例示例一样——必须使用参数化函数近似方案。无论是线性的还是非线性的，函数逼近都依赖于一些特性，这些特性必须易于学习系统获得，并且能够传递熟练性能所必需的信息。强化学习的大多数成功应用，在很大程度上都归功于基于人类知识和对要解决的特定问题的直觉而精心设计的一系列特性。谷歌 DeepMind 的一组研究人员开发了一个令人印象深刻的演示，即一个深层多层 ANN 可以自动完成特性设计过程 (Mnih et al.，2013, 2015)。自 1986 年将反向传播算法推广为学习内部表示方法以来，多层 ANNs 被用于增强学习中的函数逼近 (Rumelhart, Hinton, Williams, 1986); 见 9.6 节)。将强化学习与反推结合，得到了显著的效果。特索罗和同事们用 TD-Gammon 和 Watson 得到的结果是值得注意的例子。这些应用程序和其他应用程序得益于多层 ANNs 学习任务相关特性的能力。然而，在我们所知道的所有示例中，最令人印象深刻的演示要求网络的输入以针对给定问题手工制作的特殊特性来表示。这在 TD-Gammon 结果中表现得很明显。TD-Gammon 0.0，其网络输入本质上是西洋双陆棋棋盘上的“原始”表示，这意味着它几乎不涉及西洋双陆棋的知识，因此也学会了近似地玩西洋双陆棋

以前最好的西洋双陆棋计算机程序。添加专门的 backgammon 特性产生了 TD-Gammon 1.0，它比以前所有的 backgammon 程序都要好，并且与人类专家的竞争也很好。Mnih 等人开发了一种名为 deep Q-network (DQN) 的增强学习代理，它将 Q-learning 与深度卷积神经网络 (deep tional ANN) 结合在一起。我们在 9.6 节中描述了深层的卷积神经网络。在 Mnih 等人对 DQN 的研究中，深层 ANNs，包括深层卷积 ANNs，在很多应用中都产生了令人印象深刻的结果，但它们并没有被广泛应用于强化学习中。

Mnih 等人使用 DQN 来展示增强学习代理如何在不使用不同问题的特性集的情况下，在不同问题的集合中获得高水平的性能。为了证明这一点，他们让 DQN 通过与游戏模拟器交互来学习玩 49 个不同的 Atari 2600 视频游戏。DQN 学到了不同的政策的 49 场比赛 (因为安被重置为随机值的权重在学习每个游戏), 但它使用相同的原始输入, 网络体系结构, 参数值 (如步长, 折现率, 探索参数, 以及更多特定于实现) 的游戏。在这些游戏中，DQN 在很大程度上达到或超越了人类水平。虽然这些游戏都是通过观看视频图像来玩的，但它们在其他方面有很大的不同。他们的行为有不同的效果，他们有不同的状态转换动力，他们需要不同的政策来学习高分。深度卷积神经网络学会了将所有游戏的原始输入转换为特征，以表示在大多

数游戏中达到的高级 DQN 所需的动作值。

雅达利 2600 是一款家庭视频游戏机，从 1977 年到 1992 年，雅达利公司 (Atari Inc.) 一直在销售各种版本的游戏机。它引入或推广了许多现在被认为是经典的街机电子游戏，比如 Pong, Breakout, Space 入侵者，和小行星。虽然比现代电子游戏简单得多，但雅达利 2600 游戏对人类玩家来说仍然是娱乐和具有挑战性的，而且它们已经成为开发和评估强化学习方法的试验台 (Diuk, Cohen, Littman, 2008;Naddaf,2010;Cobo, Zang, Isbell 和 Thomaz, 2011; 贝勒玛尔,《活力》和《保龄球》, 2013 年)。Bellemare、Naddaf、Veness 和 Bowling(2012) 开发了公开的街机学习环境 (ALE)，以鼓励和简化使用 Atari 2600 游戏来学习和规划算法。这些先前的研究和 ALE 的可用性使 Atari2600 游戏集成为 Mnih 等人演示的一个很好的选择，这也受到 TD-Gammon 在 backgammon 中令人印象深刻的人类水平表现的影响。DQN 类似于 TD-gammon，使用多层 ANN 作为 TD 算法的半梯度形式的函数逼近方法，梯度由反向传播算法计算。然而, 而不是使用 TD( )TD-Gammon 一样,DQN semi-gradient 的 q 学习的形式使用。TD-Gammon 估计后态的值，很容易从《西洋双陆棋规则》中得到。为 Atari 游戏使用相同的算法，需要为每个可能的操作生成下一个状态 (在这种情况下，这是不可能的)。

这可以通过使用游戏仿真器对所有可能的操作 (ALE 使之成为可能) 运行单步模拟来实现。或者每个游戏的状态转换函数的模型可以被学习并用于预测下一个状态 (Oh, Guo, Lee, Lewis, and Singh, 2015)。虽然这些方法可能产生与 DQN 类似的结果，但它们的实现会更加复杂，并且会显著增加学习所需的时间。使用 Q-learning 的另一个动机是，DQN 使用了如下所述的经验重播方法，这需要一个非策略算法。无模型和偏离策略使 Q-learning 成为自然选择。在描述 DQN 的细节和如何进行实验之前，我们先看看 DQN 能够达到的技能水平。Mnih 等人将 DQN 的分数与当时文献中表现最好的学习系统的分数、一个专业的人类游戏测试者的分数、一个随机选择行为的代理人的分数进行了比较。来自文献的最好的系统使用了线性函数近似的特性，利用一些关于 Atari 2600 游戏的知识 (Bellemare, Naddaf，, and Bowling, 2013)。DQN 通过与游戏模拟器进行 5000 万帧的交互来学习每一款游戏，相当于 38 天的游戏体验。在每个游戏开始学习时，DQN 网络的权值被重置为随机值。为了评估 DQN 在学习后的技能水平，每一场比赛的平均得分超过 30 次，每次持续 5 分钟，开始于随机初始状态。专业的人类测试人员使用相同的模拟器 (声音关闭，以删除不处理音频的 DQN 的任何可能的优势)。经过 2 个小时的训练，人类在每个游戏中播放大约 20 集，每次最多 5 分钟，在此期间不允许休息。DQN 在所有游戏中都比之前最好的强化学习系统玩得更好，除了 6 个，并且在 22 个游戏中比人类玩家玩得更好。Mnih 等人认为，任何在人类得分或超过 75% 以上的成绩都可以与人类水平的比赛相媲美，或者比人类水平的表现更好，这一结果表明，在 46 场比赛中的 29 场比赛中，DQN 的得分达到或超过了人类的水平。Mnih 等人 (2015) 对这些结果有更详细的描述。人工学习系统来实现这些水平的发挥将足够令人印象深刻, 但是让这些结果引人注目, 许多当时被认为是人工智能是突破性的结果, 同样的学习系统实现这些水平的不同游戏不依赖任何游戏的修改。

人类这些 49 雅达利游戏看到 210×160 像素图像帧与 128 颜色 60 赫兹。原则上, 到底这些图像可能形成 DQN 原始输入, 但减少内存和处理要求,Mnih 等人预处理每一帧产生 84×84 的亮度值。因为许多雅达利的游戏的完整状态不完全可观测的图像帧,Mnih et al。“堆叠”最近的四帧, 以便输入到网络尺寸 84×84×4。这并没有消除所有游戏的部分可观测性，但它有助于使其中许多游戏更具有马尔可夫性。这里关键的一点是，这些预处理步骤对于所有 46 个都是完全相同的

游戏。除了一般的理解之外，没有涉及特定于游戏的先验知识，即仍然有可能通过这种减少的维度学习到良好的策略，并且叠加相邻帧应该有助于某些游戏的部分可观察性。因为没有游戏先验知识除此之外最少用于预处理图像帧, 我们能想到的 84×84×4 输入向量作为“原始”DQN 的输入。DQN 的基本架构类似于图 9.15 中所示的深层卷积神经网络 (尽管与该网络不同，DQN 中的子采样被视为每个卷积层的一部分，其中的特征映射由只选择可能接受域的单元组成)。DQN 有三个隐藏的卷积层，然后是一个完全连接的隐藏层，然后是输出层。DQN 产生的三个连续卷积隐藏层 32 20×20 特征图,64 9×9 特征图,64 7×7 特征图。每个

feature map 的单位的激活函数是一个整流器非线性 (max(0,x))。3136(64×7×7) 单位在这个第三卷积层所有连接到每个 512 单位的完全连接隐藏层, 然后每个连接到输出层中的所有 18 个单位, 每个可能的行动在雅达利一个游戏。DQN 输出单元的激活水平为相应状态-动作对的估计最优动作值，以网络输入为代表。游戏动作的输出单位的分配因游戏的不同而不同，由于游戏的有效动作数量在 4 到 18 之间不同，并不是所有的输出单位在所有游戏中都有功能角色。把网络想象成 18 个独立的网络，一个用来估计每个可能的行动的最佳行动值。实际上，这些网络共享它们的初始层，但是输出单元学会了使用这些层以不同的方式提取的特性。

DQN 奖赏信号显示游戏的分数如何改变从一个时间到下一个步:+ 1 时增加,−1 每当它减少, 否则和 0。这使整个游戏的奖励信号标准化，并使一个单步大小的参数在所有游戏中都有效，尽管它们的分数范围不同。DQN -greedy 政策, 与 减少线性第一个百万帧和保持在一个低价值的学习会议。通过进行非正式的搜索，我们选择了各种其他参数的值，比如学习步长、贴现率和其他特定于实现的参数，以查看哪些值最适合于少量的游戏。这些价值观在所有的比赛中都是固定的。DQN 选择一个动作后，该动作由游戏模拟器执行，该模拟器返回一个奖励和下一个视频帧。帧被预处理并添加到作为网络下一个输入的四帧堆栈中。暂时跳过 Mnih 等人对基本 Q-learning 过程的修改，DQN 使用 Q-learning 的以下半梯度形式更新网络的权值:

wt + 1 = wt +
Rt + 1 + max 一个问ˆ(圣 + 1,wt)− 问ˆ(圣, 在 wt)
wt 问ˆ(圣),(16.3)

其中 wt 为网络权值向量，At 为时间步 t 选取的动作，St 和 St+1 分别为时间步 t 和 t+1 对网络进行预处理的图像堆栈输入。(16.3) 的梯度是通过反向传播计算的。再想象,

对于每个动作都是一个单独的网络，对于 t 时刻的更新，反向传播只应用于与 at 对应的网络。Mnih 等人在应用于大型网络时，利用显示的技术改进了基本的反向传播算法。他们使用了一个小批处理方法，在一小批图像上 (这里是 32 张图像之后) 积累梯度信息后，才更新权重。与在每次操作之后更新权重的通常过程相比，这产生了更平滑的示例梯度。他们还使用了一种名为 RMSProp (Tieleman 和 Hinton, 2012) 的梯度上升算法，它通过调整每个权重的步长参数来加速学习，这是基于该权重的最近梯度的运行平均值。

Mnih 等人以三种方式修改了基本的 Q-learning 过程。首先，他们使用了 Lin(1992) 首先研究的经验重播方法。该方法将代理的经验存储在重播内存中的每个时间步骤中，重播内存被访问以执行重量更新。它在 DQN 中是这样工作的。游戏仿真器以图像堆栈 St 表示的状态执行动作后，返回奖励 Rt+1 和图像堆栈 St+1，将 tuple (St, At, Rt+1, St+1) 添加到重播内存中。这种记忆在同一游戏的许多游戏中积累了经验。在每次步骤中，根据从重播内存中随机抽取的经验，执行多个 Q-learning 更新 (一个微型批处理)。不像通常的 Q-learning 方式那样，St+1 成为下一个更新的新 St，而是从重播内存中提取一种新的无连接体验，为下一次更新提供数据。由于 Q-learning 是一种非策略算法，因此不需要沿着连通轨迹进行应用。

与传统的 Q-learning 方式相比，带经验重播的 Q-learning 提供了一些优势。能够将每个存储的经验用于许多更新，使 DQN 能够更有效地从经验中学习。体验重播降低了更新的方差，因为连续的更新彼此之间没有相关性，就像标准 Q-learning 一样。通过消除连续经验对当前权重的依赖，经验重复消除了不稳定性的一个来源。

Mnih 等改进了标准 Q-learning，以提高其稳定性。与其他引导方法一样，Q-learning 更新的目标依赖于当前的动作-值函数估计。当使用参数化函数逼近方法来表示操作值时，目标是正在更新的相同参数的函数。例如, 更新的目标 (16.3) 给出的是 maxa q̂(圣 + 1,wt)。它对 wt 的依赖与目标不依赖于被更新的参数的更简单的监督学习情况相比，使过程更加复杂。正如第 11 章所讨论的那样，这会导致振荡和/或散度。

为了解决这个问题，Mnih 等人使用了一种技术，使 Q-learning 更接近于更简单的监督学习案例，同时仍然允许它自我引导。每当一定数量,C, 更新所做的重量 w 行为价值网络, 他们将网络目前的重量到另一个网络, 这些重复的权重固定 C 更新 (w)。这个重复的输出网络未来 C 更新 (w) 被用作 q 学习的目标。让问ˇ 表示这个复制网络的输出, 然后代替 (16.3) 更新

规则是:

wt + 1 = wt +

Rt + 1 + max 一个问˜(圣 + 1,wt)− 问ˆ(圣, 在 wt)

wt 问ˆ(St)。

最后对标准 Q-learning 进行了修改，以提高稳定性。他们剪了误差项 Rt + 1 + maxa 问˜(圣 + 1,wt)− 问ˆ(圣, 在 wt) 因此留在间隔 (−1,1)。Mnih 等人对其中的 5 个游戏进行了大量的学习，以深入了解 DQN 的各种设计特性对其性能的影响。他们运行 DQN，包含或不包含四种体验重播组合和重复目标网络。尽管从游戏到游戏的结果各不相同，但单独的这些特性都显著地提高了性能，并且在一起使用时，性能得到了极大的提高。Mnih 等人还通过比较 DQN 的深度卷积版本和一个只有一个线性层的网络的版本 (两者都接收相同的叠置预处理视频帧) 来研究深度卷积神经网络在 DQN 学习能力中的作用。在这里，深度卷积版本比线性版本的改进在所有 5 个测试游戏中尤其引人注目。

人工智能的一个长期目标是创造出一种能够在各种挑战性任务中脱颖而出的人工智能。机器学习作为实现这一目标的一种方式的承诺，由于需要解决特定问题的表述而受挫。DeepMind 的 DQN 是向前迈出的重要一步，它证明单个代理可以学习问题特定的特性，从而使它能够在一系列任务中获得人类竞争技能。这个演示并没有产生一个同时擅长所有任务的代理 (因为每个任务都是单独学习的)，但是它表明深度学习可以减少、甚至可能消除针对问题的设计和调优的需求。然而，正如 Mnih 等指出的那样，DQN 并不能完全解决任务自主学习的问题。虽然要在雅达利游戏中表现出色所需要的技能是多种多样的，但所有的游戏都是通过观察视频图像来进行的，这使得深度卷积神经网络成为完成这些任务的自然选择。此外，DQN 在一些雅达利 2600 游戏上的表现也大大低于这些游戏的人类技能水平。对于 DQN 来说，最困难的游戏——尤其是 DQN 学会了如何像随机玩家一样进行报复的 Montezuma 的游戏——需要进行比 DQN 设计的更深入的规划。此外，通过广泛的实践来学习控制技能，就像 DQN 学会了如何玩雅达利游戏一样，这只是人类常规完成的学习类型之一。尽管存在这些局限性，DQN 在机器学习方面取得了长足的进步，但却令人印象深刻地展示了将强化学习与现代深度学习方法相结合的承诺。

15.6 掌握围棋

几十年来，中国古代围棋一直挑战着人工智能研究者。在其他游戏中获得人类水平技能，甚至超人类水平技能的方法，在生成强大的围棋程序方面并不成功。由于围棋程序员和国际比赛的活跃，围棋程序的水平在过去几年中有了显著的提高。然而，直到最近，还没有围棋程序能达到人类围棋大师的水平。DeepMind (Silver et al.，2016) 的一个团队开发了 AlphaGo 程序，该程序已经崩溃

这种障碍结合了深度 ANNs(第 9.6 节)、监督学习、蒙特卡罗树搜索 (MCTS，第 8.11 节) 和强化学习。在 Silver 等人 2016 年出版之前，AlphaGo 已经被证明比目前其他围棋程序更强大，并以 5 比 0 击败了欧洲围棋冠军樊麾。这是围棋程序第一次在围棋游戏中击败专业的人类棋手而没有任何障碍。此后不久，类似的阿尔法狗击败了 18 届世界冠军李世石 (Lee Sedol)，在 5 场挑战赛中赢得 4 场，成为全球新闻头条。人工智能研究人员认为，一个程序要达到这样的水平还需要许多年甚至几十年的时间。

这里我们描述 AlphaGo 和一个叫做 AlphaGo Zero (Silver et al. 2017a) 的后续程序。除了强化学习之外，AlphaGo 还依赖于从一个大型人类动作专家数据库中进行的监督学习，而 AlphaGo Zero 只使用强化学习，除了游戏的基本规则之外，不使用任何人类数据或指导 (因此得名“零”)。我们首先对 AlphaGo 进行了一些详细的描述，以突出 AlphaGo Zero 的相对简单性。

在很多方面，AlphaGo 和 AlphaGo Zero 都是 Tesauro 的 TD-Gammon(第 16.1 节) 的后代 (第 16.2 节)。所有这些项目都包括强化学习，而不是模拟的自我游戏。AlphaGo 和 AlphaGo

Zero 也建立在 DeepMind 使用 DQN 程序 (第 16.5 节) 进行 Atari 游戏的进展上，该程序使用深度卷积 ANNs 来近似最优值函数。

一试板配置围棋是两名棋手之间的一种游戏，他们交替地把黑和白的“石头”放在未占用的十字路口或“点”上，棋盘上有 19 条横线和 19 条垂线，以产生右边显示的位置。游戏的目标是捕获比对手捕获的面积更大的区域。石头是根据简单的规则捕获的。如果一个玩家的石头完全被另一个玩家的石头包围，那么他的石头就会被捕获，这意味着没有水平或垂直的相邻点没有被占用。例如，图 16.5 显示了左边的三颗白色石头，旁边有一个未被占用的点 (标记为 X)。然而, 如果玩家 white 首先在 X 点上放置一块石头，那么这个捕获的可能性将被阻止 (图 16.5 右边)。需要其他规则来防止无限捕获/重新捕获循环。当双方都不愿再放一块石头时，游戏就结束了。这些规则很简单，但是它们产生了一种非常复杂的游戏，而且非常广泛

图 16.5:Go capture 规则。左图: 这三颗白色的石头并没有因为点而被包围 X 是空置的。中间: 如果黑色在 X 上放置一块石头，三个白色的石头就会被捕获退出董事会。正确: 如果白色首先在 X 点放置一块石头，则捕获被阻塞。

上千年的呼吁。在围棋等其他游戏中发挥强大作用的方法，在围棋中表现得并不好。搜索空间去显著大于国际象棋, 因为有更多的法律行动/位置比国际象棋 (250 和 35), 游戏往往涉及移动比国际象棋游戏 (150 与 80)。但是，搜索空间的大小并不是导致搜索变得如此困难的主要因素。详尽的搜索是不可行的国际象棋, 和继续小板 (如 9×9) 已被证明是非常困难的。专家们一致认为，创建比业余围棋更强的围棋程序的主要障碍是很难定义一个合适的位置评估函数。一个好的评估函数允许搜索在一个可行的深度上被截断，通过提供相对容易计算的关于深度搜索可能产生的结果的预测。根据 ¨ll(2002): “不简单而合理的评价函数会被发现。一个重要的进步是引入了 MCTS 来进行程序。AlphaGo 开发时最强大的程序都包括 MCTS，但大师级别的技能仍然难以捉摸。

回想一下第 8.11 节，MCTS 是一个决策时规划过程，它不尝试学习和存储全局评估函数。就像推出算法 (第 8.10 节) 一样，它运行许多蒙特卡罗模拟整个场景 (在这里，整个围棋游戏) 来选择每个动作 (这里，每个动作: 在哪里放置石头或辞职)。然而，与简单的 rollout 算法不同，MCTS 是一个迭代过程，它递增地扩展搜索树，其根节点表示当前环境状态。如图 8.10 所示，每个迭代通过模拟与树的边缘相关的统计信息指导的操作来遍历树。在其基本版本中，当模拟到达搜索树的叶子节点时，MCTS 通过向树中添加一些或全部的叶子节点子节点来扩展树。从叶子节点或新添加的子节点中执行一个 rollout: 一个通常一直执行到终端状态的模拟，其操作由 rollout 策略选择。当 rollout 完成时，与在此迭代中遍历的搜索树的边缘相关的统计信息将通过备份 rollout 产生的返回来更新。MCTS 继续这个过程，每次从搜索树的根开始，在当前状态，在给定时间限制的情况下，尽可能多地进行迭代。然后，最后，根据根节点的传出边中累积的统计信息，选择根节点 (仍然表示当前环境状态) 的操作。这是代理所采取的行动。在环境转换到下一个状态之后，MCTS 再次被执行，根节点集将代表新的当前状态。开始的搜索树

下一个执行可能仅仅是这个新的根节点，也可能包含 MCTS 以前执行时遗留下来的这个节点的后代。树的其余部分被丢弃。

15.6.1 AlphaGo

AlphaGo 之所以成为如此强大的棋手，其主要创新之处在于，它采用了一种新型的 MCTS 棋类，这种棋类既遵循一种策略，又遵循一种价值函数，通过深度卷积神经网络提供的函数逼近进行强化学习。另一个关键特性是，它不是从随机的网络权重开始强化学习，而是从之前大量的人类专家动作的监督学习的结果开始。DeepMind 团队将 AlphaGo 对基本 MCTS 的修改称为“异步策略和值 MCTS”，即 APV-MCTS。它通过上面描述的基本的 MCTS 来选择动作，但是在如何扩展搜索树以及如何评估动作边缘方面有一些变化。基本 mct 相比, 使用存储扩展当前的搜索树的行动值从一个叶节点选择一个未知的边缘,APV-MCTS, 作为 AlphaGo 实现, 扩大了选择一条边的树根据概率由 13-layer 深安卷积, 称为 SL-policy 网络, 由监督学

习训练之前预测移动数据库中包含近 3000 万人类专家的动作。

然后, 还与基本 mct, 评估新增状态返回的节点完全推出发起,APV-MCTS 评估节点在两个方面: 这个返回的推广, 但也由价值函数,v , 学会了以前的强化学习方法。如果 s 是新添加的节点，它的值就变成了。

v(s)=(1−)v (s)+ G, (16.4)

G 是推出的回归和 值的混合控制造成这两种评价方法。在 AlphaGo 中，这些值由值网络提供，这是另一个 13 层的深度卷积神经网络，我们将其训练为输出板位的估计值。APV-MCTS 在 AlphaGo 中进行了模拟游戏，两名玩家都使用一个简单的线性网络提供的快速推出策略，在游戏开始前还接受了监督学习的训练。在整个执行过程中，APV-MCTS 记录了在搜索树的每个边缘通过了多少次模拟，当它的执行完成时，从根节点获得最多访问的边缘被选择为要采取的行动，在这里，AlphaGo 实际上是在一个游戏中移动的。

值网络与深度卷积 SL 策略网络具有相同的结构，只是它有一个输出单元，提供游戏位置的估计值，而不是 SL 策略网络对法律行为的概率分布。理想情况下, 网络输出值最优状态值, 它可能是可能的近似最优值函数的 TD-Gammon 上面描述:self-play 与 TD( ) 耦合的非线性卷积安。但是 DeepMind 团队采用了一种不同的方法，对于像 Go 这样复杂的游戏来说，这种方法更有希望。他们把培训价值网络的过程分为两个阶段。在第一阶段，他们通过使用强化学习来训练一个 RL，创造了他们能做的最好的策略

政策网络。这是一个与 SL 策略网络结构相同的深度卷积神经网络。通过监督学习得到的 SL 策略网络的最终权重初始化，然后使用策略梯度强化学习对 SL 策略进行改进。在价值网络训练的第二阶段，团队使用蒙特卡罗策略评价方法，对 RL 策略网络所选取的大量模拟自玩游戏的数据进行评价。图 16.6 展示了 AlphaGo 使用的网络，以及在 DeepMind 团队所称的“AlphaGo 管道”中训练它们的步骤。“所有这些网络都是在现场比赛开始前进行训练的，他们的重量在整个现场比赛中都是固定的。”

人类专家的职位 Self-play 位置

图 16.6:AlphaGo 管道。改编自麦克米伦出版社有限公司: 自然，第 529 卷 (7587 页)，第 485 页，c ?2016 年。

这里有一些关于 AlphaGo 的 ANNs 和他们的训练的细节。identically-structured SL 和 RL 政策网络类似于 DQN 深卷积网络 16.5 节中描述的雅达利游戏, 除了他们有 13 卷积层和最后一层组成的 soft-max 单位为每个点 19×19 板。网络的输入是一个 19×19×48 图像栈中每个点的董事会由 48 个二进制或整数值特性的值。例如，对于每一个点，一个特性表明这个点是被 AlphaGo 的一颗石头 (对手的石头之一) 占据的，还是没有占用的，从而提供了棋盘配置的“原始”表示。其他功能都是基于规则的, 如相邻的点的数量是空的, 对手石头的数量将被放置一块石头, 石头以来匝数是放置在那里, 和其他特性, 设计团队认为是重要的。使用 50 个处理器上的随机梯度提升的分布式实现，训练 SL 策略网络大约需要 3 周的时间。该网络的准确率达到 57%，这是其他组在出版时达到的最佳准确率

是 44.4%。RL 策略网络的训练是在 RL 策略网络的当前策略与对手之间的模拟博弈中，通过策略梯度强化学习，从学习算法早期迭代生成的策略中随机选择策略。与随机挑选的反对者进行比赛，避免了对现行政策的过度配合。奖励信号 + 1 如果当前政策赢得 −1 如果它丢失, 否则为 0。这些游戏直接将两种政策对立起来，而不涉及 MCTS。DeepMind 团队通过在 50 个处理器上同时模拟许多游戏，在一天内训练了 100 万场游戏的 RL 策略网络。在测试最终的 RL 政策时，他们发现它赢得了超过 80% 的游戏，而在 SL 政策中，它赢得了 85% 的游戏，而使用 MCTS，模拟了每移动 10 万次游戏。值网络的结构与除了单一输出单元外的 SL 和 RL 策略网络相似，但接收到与 SL 和 RL 策略网络相同的输入，但有一个附加的二进制特性可以显示当前的颜色。蒙特卡罗政策评估是利用 RL 策略从大量的自玩游戏中获得的数据来训练网络的。为了避免过度拟合和不稳定性，DeepMind 团队构建了一个数据集，每个数据集都是从一个独特的自玩游戏中随机选择的 3000 万个位置。然后用 5000 万次小批量的训练，每个小批量的训练都是从这个数据集中抽取的 32 个位置。

在一个简单的线性网络训练之前，我们学习了一个简单的线性网络，这个网络是由一个

800 万人的移动的语体训练出来的。推出策略网络必须在输出动作迅速的同时仍然保持合理的准确性。原则上,SL 或 RL 政策网络可以用于糊涂事, 但远期通过这些深层网络传播他们花了太多时间在推广使用模拟, 其中许多为每个移动决定在进行生活。由于这个原因，rollout 策略网络没有其他策略网络那么复杂，它的输入特性可以比用于策略网络的特性更快地计算出来。rollout 策略网络允许每秒在 AlphaGo 使用的每个处理线程上运行大约 1000 次完整的游戏模拟。人们可能想知道为什么在 APV-MCTS 的扩展阶段使用 SL 策略而不是更好的 RL 策略来选择操作。这些策略的计算时间相同，因为它们使用相同的网络架构。研究团队发现，当 APV-MCTS 使用 SL 策略而不是 RL 策略时，AlphaGo 在对抗人类对手时表现得更好。他们推测，原因是后者是为了响应最优的动作而不是人类游戏的更广泛的动作。有趣的是，APV-MCTS 使用的值函数的情况正好相反。他们发现当 APV-MCTS 使用从 RL 策略派生的值函数时，它的性能比使用从 SL 策略派生的值函数要好。

几种方法共同作用，产生了 AlphaGo 令人印象深刻的演奏技巧。DeepMind 团队评估了不同版本的 AlphaGo，以评估其贡献。

由这些不同的部件组成。参数 (16.4) 的混合控制网络和游戏状态评估产生的价值糊涂事。 = 0,AlphaGo 使用价值网络没有糊涂事, = 1, 评价只是依赖糊涂事。他们发现，只使用值网络的 AlphaGo 比只使用 rollout 的 AlphaGo 表现得更好，实际上也比当时所有强大的围棋程序表现得更好。最好的发挥造成设置 = 0.5, 表明结合价值网络与电 AlphaGo 的成功尤为重要。这些评价方法相互补充: 价值网络太慢的高性能 RL 政策评估用于生活, 虽然糊涂事使用较弱但更快推出政策能够添加价值网络的精度评估的特定状态发生在游戏。总的来说，AlphaGo 的非凡成功激起了新一轮的热情，人们对人工智能的前景充满了热情，特别是对于将强化学习与深度人工智能相结合的系统，以解决其他具有挑战性领域的问题。

15.6.2 AlphaGo 零

基于 AlphaGo 的经验，DeepMind 团队开发了 AlphaGo Zero (Silver et al. 2017a)。与 AlphaGo 不同的是，这个程序除了游戏的基本规则外，没有使用任何人类数据或指导 (因此名称为 0)。它只从自玩强化学习中学习，输入只给出对围棋棋盘上的石头位置的“原始”描述。AlphaGo Zero 实现了一种形式的策略迭代 (第 4.3 节)，将策略评估与策略改进交叉。图 16.7 是 AlphaGo Zero 算法的概述。AlphaGo 0 和 AlphaGo 的一个显著区别是，AlphaGo Zero 使用 mct 在自玩强化学习中选择移动，而 AlphaGo 则使用 MCTS 进行实时游戏，而不是持续学习。除了不使用任何人工数据或人工设计的特性之外，AlphaGo Zero 还存在其他的不同，即它只使用了一个深度卷积神经网络，并使用了一个更简单的 MCTS 版本。

AlphaGo Zero 的 MCTS 比 AlphaGo 使用的版本更简单，因为它不包含完整游戏的推出，因此不需要推出策略。AlphaGo Zero 的 MCTS 每一次迭代都运行一个模拟，最后在当前搜索树的叶子节点上结束，而不是在完整的游戏模拟的终端位置。但 AlphaGo, 每次迭代 AlphaGo 零 mct 的指导下的输出卷积网络深处, 贴上 f 在图 16.7 中, 是网络的权向量。网络的输入, 其架构下面我们描述, 包括董事会的原始表示位置, 及其输出有两个部分: 一个标量值,v, 估计当前的球员会赢的概率从现任董事会的位置, 和一个矢量,p, 转移概率, 每个可能的石头放置在当前一个板, 加上通过, 或者辞职, 那就动起来吧。而不是选择 self-play 行动根据概率 p, 然而,AlphaGo 零概率使用, 加上网络的价值输出, 直接每个特定的执行, 返回新的移动概率, 政策 i 如图 16.7 所示。这些政策受益于 MCTS 所做的许多模拟。

图 16.7:AlphaGo 零级自我强化学习。a) 程序对自己进行了许多游戏，这里显示的一个是棋盘位置序列 si, i = 1,2，…T, 移动 ai i = 1 2…,T, 赢家 z。每个移动概率 i ai 是由行动从根节点返回的特定执行 si 和指导下深卷积网络, 在这里贴上 f , 最新的重量 。这里只显示了一个位置 s，但对所有 si 都是重复的，网络的输入是板位置 si 的原始表示 (连同几个过去的位置，虽然这里没有显示)，它的输出是移动概率的向量 p 这指导了 MCTS 的正向搜索，标量值 v 估计了当前玩家从每个位置 si 中获胜的概率。b) 深度卷积网络训练。培训我们从最近的自玩游戏中随机抽取了一些例子。权重 更新将政策矢量 p 对 mct 返回的概率 , 包括估计

的赢家 z 赢得概率诉转载草案的银 et al。(2017) 经作者许可和 DeepMind。

每次执行。结果是，AlphaGo Zero 实际上遵循的策略比网络输出 p. Silver 等人 (2017a) 给出的策略有所改进，“因此，MCTS 可能被视为一个强大的策略改进操作符。”

下面是关于 AlphaGo Zero 的 ANN 的更多细节，以及它是如何训练的。网络作为输入了 19×19×17 飞机 17 二进制组成的图像叠加特性。前 8 个特征面是当前玩家的石头在当前和过去 7 个棋盘配置中的原始表示: 如果玩家的石头在相应的点上，特征值为 1，否则为 0。接下来的 8 个特征面类似地编码了对手石头的位置。最后一个输入特征平面具有一个常量值，该值指示当前播放的颜色:1 表示黑色;0 为白色。因为围棋不允许重复，一个棋手因为没有得到第一步棋而被给予一定的“补偿点数”，所以当前棋盘的位置不是围棋的马尔可夫状态。这就是为什么需要描述过去董事会位置和颜色特征的特征。

网络是“双头”的，这意味着在经过了若干初始层之后，网络被分成了两个独立的“头”层，这些“头”层又被分成了两组输出单元。在这种情况下，一个人头输入 362 个输出单元，产生 192 + 1 的移动概率 p，每一个可能的石头放置加通过; 另一个头只给一个输出单元输入标量 v，这是当前玩家从当前棋盘位置获胜的概率的估计值。分割之前的网络由 41 个卷积层组成，每个层之后都进行批处理规范化，并通过添加跳过连接实现对层的剩余学习 (参见 9.6 节)。总的来说，移动概率和数值分别由 43 层和 44 层计算。

从随机权重开始，该网络通过随机梯度下降 (随着训练的进行，动量、正则化和步长参数逐渐减小) 对其进行训练，使用随机抽样的样本，这些样本来自最近 50 万场自玩游戏的所有步骤。额外的噪音被添加到网络的输出 p，以鼓励探索所有可能的移动。西尔弗等人 (2017a) 在培训期间的定期检查点 (时选择每 1000 个培训步骤) 中，使用最新权重的 ANN 对当前最佳策略进行模拟 400 个游戏 (使用 1600 次迭代的 MCTS 选择每个动作) 来评估策略输出。如果新政策赢得了 (以降低结果的噪音为目标)，那么它就成为了在随后的自演中使用的最佳政策。更新网络的权值，使网络的策略输出 p 更接近 MCTS 返回的策略，使其值输出 v 更接近于当前最佳策略从网络输入所代表的棋盘位置获胜的概率。

DeepMind 团队对 AlphaGo Zero 进行了 490 多万次自我游戏的训练，耗时约 3 天。每个游戏的每次移动都是通过运行 MCTS 进行 1600 次迭代来选择的，每次移动大约需要 0.4 秒。网络重量更新超过 70 万批次，每批包含 2048 块板配置。然后，他们与训练有素的 AlphaGo Zero 进行比赛，与 AlphaGo 版本的比分是 5 比 0，击败了樊麾; 与 AlphaGo 版本的比分是 4 比 1，击败了李世石。他们使用 Elo 评级系统来评估项目的相对表现。两个 Elo 评分之间的差异是为了预测玩家之间游戏的结果。AlphaGo Zero (AlphaGo Zero) 和李世石 (Lee Sedol) 的 Elo 评分分别为 4,308、3,144 和 3,739。Elo 评级的差距转化为 AlphaGo 0 击败其他程序的概率非常接近于 1 的预测。在 100 场比赛中，AlphaGo 0 和击败李世石 (Lee Sedol) 的 AlphaGo 的确切版本在 100 场比赛中都以相同的条件击败了 AlphaGo。

DeepMind 团队还将 AlphaGo Zero 与一个程序进行了比较，该程序使用具有相同结构的人工神经网络 (ANN)，但经过监督学习 (supervised learning) 训练，在一个包含 16 万场游戏近 3000 万个位置的数据集中预测人类的移动。他们发现，最初 AlphaGo 的表现比 AlphaGo Zero 要好，而且更善于预测人类专家的动作，但在 AlphaGo Zero 接受一天训练后，他们的表现就不那么好了。这表明，AlphaGo Zero 已经发现了一种游戏策略。

不同于人类的游戏方式。事实上，AlphaGo Zero 发现并开始偏爱一些经典移动序列的新变种。

AlphaGo Zero 算法的最终测试是在一个拥有更大的 ANN 的版本中进行的，并且训练了超过 2900 万的自玩游戏，这需要 40 天的时间，再一次从随机的重量开始。这个版本达到了 5 185 的 Elo 评级。研究小组将这一版本的 AlphaGo Zero 与当时最强大的 AlphaGo Master 程序进行了对比，该程序与 AlphaGo Zero 完全相同，但与 AlphaGo 一样，它使用了人类数据和功能。AlphaGo Master 的 Elo 评分为 4858，在网络游戏中以 60 比 0 击败了人类最强大的职业棋手。在 100 场比赛中，拥有更大网络和更广泛学习的 AlphaGo Zero 以 11 比 89 击败了 AlphaGo 大师，从而令人信服地证明了 AlphaGo Zero 的算法解决问题的能力。

AlphaGo Zero 充分证明了通过纯粹的强化学习、简单的 MCTS 版本增强、深入的 ANNs

对领域的了解非常少、不依赖人类数据或指导就可以实现超人的表现。我们肯定会看到，AlphaGo 和 AlphaGo Zero 的深度思维成就对其他领域的问题产生了启发。

最近，Silver 等人 (2017b) 描述了一个更好的程序 AlphaZero，它甚至没有包含围棋知识。AlphaZero 是一种普遍的强化学习算法，它改进了迄今为止在围棋、棋类和 shogi 等多种游戏中最优秀的程序。

15.7 个性化的 Web 服务

个性化的 web 服务，如发布新闻文章或广告，是提高用户对网站的满意度或增加营销活动的收益的一种方法。策略可以根据用户的兴趣和偏好 (从他们的在线活动历史中推断)，为每个特定用户推荐最适合他们的内容。这是机器学习的自然领域，尤其是强化学习。强化学习系统可以通过对用户反馈进行调整来改进推荐策略。获取用户反馈的一种方式是通过网站满意度调查，但为了实时获取反馈，通常会将用户点击作为链接的兴趣指标进行监控。

在市场营销中长期使用的一种叫做 A/B 测试的方法是一种简单的强化学习，用来决定网站用户喜欢的两个版本 A 或 B 中的哪个。因为它是非关联的，就像两个武装的强盗问题一样，这种方法没有个性化的内容交付。添加由描述单个用户和将要交付的内容的特性组成的上下文允许个性化服务。这已被形式化为一个上下文强盗问题 (或一个关联强化学习问题，第 2.9 节)，目标是最大化用户点击的总数。Li, Chu, Langford 和 Schapire(2010) 应用了语境的 bandit 算法来解决雅虎的个性化问题。今天的首页页面 (在他们研究的时候是互联网上访问量最大的页面之一) 通过选择新闻报道为特色。他们的目标是

最大限度地提高点击率 (CTR)，这是指所有用户在网页上点击的总次数与访问页面的总次数之比。他们的上下文土匪算法比标准的非关联土匪算法提高了 12.5%。

Theocharous, Thomas 和 Ghavamzadeh(2015) 认为，通过将个性化推荐作为一个马尔可夫决策问题 (MDP) 来实现更好的结果是可能的，目标是最大化用户在多次访问一个网站时的总点击量。从上下文土匪公式派生的策略是贪婪的，因为它们不考虑行动的长期影响。这些政策有效地对待每次访问一个网站，就好像它是由一个新访客统一抽样从网站的访客人口。由于没有利用许多用户反复访问相同网站的事实，贪婪策略没有利用与单个用户长期交互所提供的可能性。

Theocharous 等人举了一个营销策略如何利用长期用户互动的例子，将贪婪的政策与长期的政策 (比如汽车广告) 进行对比。贪婪策略显示的广告可能会提供折扣，如果用户立即购买汽车。用户要么接受要约，要么离开网站，如果他们回到网站，他们很可能会看到同样的要约。另一方面，一项长期政策可以在提交最终协议之前，将用户“从销售渠道向下转移”。它可以从描述有利的融资条件开始，然后赞扬一个优秀的服务部门，然后在下次访问时，提供最终的折扣。这种类型的策略可以导致用户在多次访问站点时产生更多的点击，如果策略设计得当，那么最终的销售就会更大。

Theocharous 等人在 Adobe 系统股份有限公司工作时，做了一些实验，看看设计用于长期最大化点击的策略是否真的能比短期的贪婪策略更好。Adobe 营销云是许多公司用来开展数字营销活动的一套工具，它提供了自动化用户广告和筹款活动的基础设施。实际上，使用这些工具部署新策略会带来重大风险，因为新策略最终可能表现不佳。由于这个原因，研究团队需要评估一个策略在实际部署时的性能，但是要基于在执行其他策略时收集的数据。这项研究的一个关键方面是偏离政策的评估。此外，该小组希望以高度的信心来降低部署新政策的风险。虽然高信任度的政策外评价是本研究的中心组成部分 (参见 Thomas, 2015;Thomas, Theocharous，和 Ghavamzadeh, 2015)，在这里我们只关注算法和它们的结果。

Theocharous 等人比较了学习广告推荐策略的两种算法的结果。第一个算法，他们称之为“贪婪优化”，它的目标是最大化立即点击的概率。与标准上下文土匪公式一样，该算法没有考虑到推荐的长期影响。另一种算法是基于 MDP 公式的增强学习算法，目的是提高用户多次访问网站的点击量。他们称之为后一种算法生命周期值 (LTV) 优化。这两种算法都面临着

挑战性的问题，因为这个领域的奖励信号非常稀疏，因为用户通常不会点击广告，而用户点击是非常随机的回报高方差。

来自银行业的数据集用于培训和测试这些算法。这些数据集包含了许多客户与银行网站互动的完整轨迹，这些轨迹显示了从可能的报价中选出的每个客户。如果顾客点击，奖励是 1，否则是 0。一组数据包含了一个银行发起的活动中大约 20 万次互动，该活动随机提供了 7 个提议中的一个。另一组数据来自另一家银行的活动，其中包含了 4,000,000 个互动，涉及 12 个可能的报价。所有的交互都包括客户特性，比如客户上次访问网站的时间、到目前为止的访问次数、客户最后一次点击的时间、地理位置、兴趣集合之一，以及提供人口统计信息的特性。

贪婪的优化是基于一个映射估计一个点击的概率作为一个用户功能的功能。这种映射是通过随机森林 (RF) 算法 (Breiman, 2001) 从一个数据集中的监督学习中获得的。RF 算法在工业上广泛应用于大型应用中，因为它们是有效的预测工具，不会过度拟合，对异常值和噪声相对不敏感。Theocharous 等人然后使用映射定义一个 -greedy 政策选择概率 1- 提供预测的 RF 算法产生点击的概率最高, 和其他选择从其他提供随机均匀。

LTV 优化使用了一种称为拟合 Q 迭代 (fit Q iteration, FQI) 的批处理模式增强学习算法。它是适合 Q-learning 的拟合值迭代 (Gordon, 1999) 的变体。批处理模式意味着学习的整个数据集从一开始就是可用的，而不是我们在本书中关注的算法的在线模式，在学习算法执行时，数据是按顺序获取的。当在线学习不实用时，批式强化学习算法有时是必要的，它们可以使用任何批式监督学习回归算法，包括已知的可扩展到高维空间的算法。FQI 的收敛取决于函数逼近算法的性质 (Gordon, 1999)。Theocharous 等人在 LTV 优化中使用了与贪婪优化方法相同的 RF 算法。因为在这种情况下 FQI 收敛不单调,Theocharous 等人记录最好的 FQI 政策 off-policy 评估使用验证训练集。最后政策测试 LTV 方法 -greedy 政策是基于最好的政策由 FQI 最初的行为价值函数集映射产生的射频贪婪优化方法。

为了度量贪婪和 LTV 方法生成的策略的性能，Theocharous 等人使用了 CTR 度量和一个他们称为 LTV 度量的度量。这些指标是相似的，除了 LTV 指标在个别网站访问者中有显著的区别: 总点击次数。访问总次数，

LTV = 总点击数总人数。

图 16.8 展示了这些度量的不同之处。每个圆圈代表用户对站点的访问; 黑圈是用户点击的访问。每一行表示一个特定用户的访问。由于没有区分访问者，这些序列的 CTR 为 0.35，而 LTV 为 1.5。因为 LTV 比 CTR 要大，以至于用户可以重新访问站点，所以它是鼓励用户与站点进行扩展交互的政策成功与否的一个指标。

图 16.8: 单击 rate (CTR) 和 life-time 值 (LTV)。每个圆圈代表用户访问; 黑圈是用户点击的访问。改编自 Theocharous 等 (2015)。

对贪婪和 LTV 方法生成的策略进行测试时，在一个测试数据集上使用了一种高度可信的离策略评估方法，该测试数据集包含与随机策略服务的银行网站的真实交互。结果表明，贪心优化在 CTR 指标下表现最佳，LTV 优化在 LTV 指标下表现最佳。此外，尽管我们忽略了它的细节，但高可信度的非政策评估方法提供了概率性的保证，即 LTV 优化方法能够在高概率的情况下制定政策，从而改进当前部署的政策。由这些概率保证担保,Adobe 宣布在 2016 年新的 LTV 算法将一个标准组件的 Adobe 营销云, 零售商可以发行后的序列提供了政策可能产生更高的回报, 而不是政策, 是对长期的结果。

15.8 热飙升

鸟类和滑翔机利用上升的气流来获得高度，以保持飞行，同时消耗很少的能量。这种行为被称为热升，是一种复杂的技能，需要对微妙的环境信号做出反应，尽可能长时间地利用上升的空气柱来增加海拔高度。Reddy、Celani、Sejnowski 和 Vergassola(2016) 利用强化学习研究了热飙升政策，这些政策在通常伴随上升气流的强大气湍流中是有效的。他们的主要目标

是提供线索，了解鸟类的感觉，以及它们如何利用它们来达到令人印象深刻的热性能，但这些结果也有助于自动滑翔机相关的技术。强化学习以前被应用于有效导航的问题

靠近热上升气流 (伍德伯里，邓恩和瓦拉瑟克，2014) 但不是更有挑战性的问题，在上升气流本身的湍流中上升。

Reddy 等人将飞涨的问题建模为持续的 MDP，并进行了贴现。这名特工与一架在湍流空气中飞行的滑翔机的详细模型进行了互动。他们投入了大量的努力，使模型产生了真实的热膨胀条件，包括研究几种不同的大气模拟方法。在学习实验中，一个有一公里边的三维盒子里的空气流动，其中一个在地面上，用一组复杂的基于物理的偏微分方程来模拟，这些偏微分方程涉及空气速度、温度和压力。在数值模拟中引入小的随机扰动，导致模型产生热上升气流模拟和伴随的湍流 (图 16.9 左)，滑翔机飞行采用气动方程建模，包括速度、升力、阻力和其他控制固定翼飞机无动力飞行的因素。操纵滑翔机需要改变它的攻击角度 (滑翔机的机翼和气流方向之间的角度) 和它的倾斜角度 (图 16.9 右侧)。

图 16.9 热升模型: 左: 垂直速度场的快照模拟空气立方体: 在红色 (蓝色) 是一个大的向上 (向下) 流动的区域。右: 无能为力飞行图显示银行角度 和攻角 。改编自《美国科学院院刊》第 113(22) 卷，第 E4879 页，2016 年，Reddy, Celani, Sejnowski 和 Vergassola，学习飞翔在动荡的环境中。

代理和环境之间的接口需要定义代理的操作、代理从环境接收的状态信息和奖励信号。通过尝试各种可能性,Reddy 等人决定三个动作每一个攻角和倾斜角足以让他们的目的: 盈亏当前银行角度和攻角 5。。和 2.5, 分别, 或者让他们改变。这导致了 32 种可能的行动。银行角度之间有界保持 −15。。+ 15。因为他们的研究的目标是试图确定有效飞翔所需要的最微小的感官线索，既要阐明鸟类可能用于飞翔的线索，又要尽量减少自动滑翔飞行所需的感知复杂性，

作者尝试了各种各样的信号作为增强学习代理的输入。他们首先使用四维状态空间的状态聚合 (第 9.3 节)，维度给出局部垂直风速，局部垂直风加速度，扭矩取决于左右翼尖垂直风速的差异，以及局部温度。每个维度被离散成三个箱子: 正高，负高，小。下面描述的结果表明，这些维度中只有两个对有效的飞翔行为至关重要。

热升的总体目标是从每一根上升的空气柱中获得尽可能多的高度。Reddy 等人尝试了一个简单的奖励信号，在每一集结束时，根据在这一集中获得的高度对特工进行奖励，如果滑翔机触地，则给出一个较大的负面奖励信号，否则为零。他们发现，在实际持续时间的片段中，利用这个奖励信号学习是不成功的，而且资格追踪也没有帮助。通过对各种奖励信号的实验，他们发现学习最好的方法是使用奖励信号，奖励信号在每个时间步上线性地结合上一个时间步上观察到的垂直风速和垂直风速加速度。

学习是一步萨尔萨 (Sarsa)，根据基于规范化行为值的软最大值分布选择动作。具体来说，动作概率是根据 (13.2) 的动作偏好来计算的: h(年代,)= 问ˆ(s,a,)−minb 问ˆ(年代,b,)

maxb 问ˆ(年代,b,)−minb 问ˆ(年代,b,)?, 是每个动作的参数向量和一个组件和聚合组, 和 q̂(s,a,) 仅仅返回组件对应的年代, 在通常状态聚合方法。上面的方程形式的行动偏好正常化的近似动作值区间 [0,1] 然后除以 , 积极“温度参数。“3 随着 的增加, 选择一个行动的概率变得不那么依赖自己的选择; 的概率, 减少向零, 选择最优先选择的操作方法, 使政策贪婪的政策。温度参数 是初始化在学习 2.0 和逐步下降到 0.2。行动偏好是计算从当前估计的行动值: 行动偏好给出的最大估计行动值是 1 / , 行动的最低估计行动值给出了偏好 0, 和其他操作的偏好是介于这两种极端情况之间。步长和折扣率参数分别固定在 0.1 和 0.98。

每一个学习过程都是由控制模拟飞行的 agent 在模拟湍流的独立生成期间进行的。每一集的时长为 2.5 分钟，每一集的时长为 1 秒。几百集之后，学习就有效地融合了。图 16.10 的左面板显示了在学习代理随机选择动作之前的样本轨迹。从所示体积的顶部开始，滑翔机的轨迹与箭头指示的方向一致，并迅速失去高度。图 16.10 右面板是学习后的轨迹。滑翔机从同一个地方开始 (这里出现在体积的底部)，通过螺旋上升获得高度

3Reddy 等人对此的描述略有不同，但我们的版本和他们的版本是一样的。

图 16.10: 热飞越轨迹样本，箭头显示从相同起始点出发的飞行方向 (注意，高度标尺被移

动)。左: 在学习之前: 代理随机选择动作，滑翔机下降。右图: 学习后: 滑翔机沿着螺旋轨迹上升高度。改编自 PNAS vol. 113(22)，p. E4879, 2016, Reddy, Celani, Sejnowski 和 Vergassola，学习在动荡的环境中飞翔。

在上升的空气中。虽然 Reddy 等人发现，在不同的模拟气流周期中，性能差异很大，但随着学习的进行，滑翔机接触地面的次数持续减少，几乎为零。在尝试了不同的特性集之后，研究人员发现，仅仅是垂直的风加速度和扭矩的组合效果最好。作者推测，由于这些特征提供了两种不同方向上的垂直风速梯度的信息，它们允许控制器通过改变银行角度来选择转弯或沿着同一路线继续前进，只留下银行角度。这使得滑翔机可以停留在上升的空气柱内。垂直风速指示了热的强度，但并不有助于保持在流中。他们发现对温度的敏感性没有什么帮助。他们还发现，控制攻击角度对保持在特定的温度下没有帮助，相反，当跨越较大的距离时，比如越野滑翔和鸟类迁徙时，它对在不同的温度之间旅行很有用。

由于不同程度的湍流需要不同的政策，所以训练在弱到强湍流的条件下进行。在强湍流中，快速变化的风和滑翔机速度使控制器的反应时间更短。这就减少了可能的控制量，而不是在波动很弱的时候。Reddy 等人研究了 Sarsa 在这些不同条件下学到的政策。在所有制度下学习到的政策的共同特点是: 当感应到负风加速时，向机翼的方向以较高的升力急速倾斜; 当感应到较大的正风加速度而无转矩时，什么也不做。然而，不同程度的动荡导致了政策上的分歧。

在强动荡中学会的政策更为保守，因为它们更喜欢小银行角度，而在弱动荡中，最好的做法是通过银行大幅转向尽可能多。系统地研究了不同条件下政策选择的银行角度，作者认为，通过检测垂直风加速度何时超过一定阈值，控制器可以调整其政策以应对不同的湍流状态。Reddy 等人也进行了实验调查贴现率的影响参数 对学习策略的性能。他们发现在一集获得的高度增加 增加, 达到最大值为 $= 0$, 表明有效热飙升需要考虑长期影响的控制决策。这一热飞的计算研究说明了强化学习如何进一步朝着不同的目标前进。学习政策可以获得不同的环境线索和控制措施，这有助于设计自动滑翔机的工程目标和提高对鸟类飞行技能的理解的科学目标。在这两种情况下，通过检测真实的滑翔机，并将预测与观察到的鸟类飞翔行为进行比较，可以在实地测试学习实验得出的假设。

15.9 第十七章前沿

在最后一章中，我们谈到了一些超出本书范围的话题，但我们认为这些话题对于强化学习的未来尤其重要。这些主题中的许多超出了可靠的已知范围，有些超出了 *MDP* 框架。

15.10 一般价值函数和辅助任务

在这本书的过程中，我们的价值函数的概念变得相当普遍。使用脱机策略学习，我们允许值函数以任意目标策略为条件。在 12.8 节我们广义打折终止函数 :S ?→[0,1], 这样不同的贴现率可以应用在每个时间步在决定返回 (12.17)。这让我们能够预测出，在一个任意的、依赖于国家的视界上，我们能得到多少回报。下一步，或许也是最后一步，是超越奖励的泛化，允许对任意信号的预测。与预测未来奖励的总和不同，我们可以预测声音或颜色感觉的未来值的总和，或者预测内部高度处理的信号，比如另一个预测。无论以这种方式在一个值函数式的预测中加入什么信号，我们都称它为该预测的累积量。我们形式化累积量信号 Ct r . 使用这种一般的价值函数, 或养狐业, 写

v , ,C(s)= E
∞ 吗?k = t 吗?k ? 我 = t + 1 (Si)
Ck + 1
圣 = s:∞
。(17.1)

与传统价值函数 (如 v 或 q∗) 这就是我们寻求一个理想函数近似的参数化形式, 我们可能会继续表示 v̂(s,w), 当然就会有不同的 w 为每个预测, 也就是说, 对于每个选择 , ,c, 因为奖励养狐业没有必然联系, 这也许是一个误称称之为价值函数。人们可以简单地称之为预测，或者更有特色地称之为预测 (圈，在准备中)。不管叫什么，它

是一个值函数的形式，因此可以用本书中介绍的学习近似值函数的方法来学习。除了习得的预测之外，我们还可以学习一些策略，以通常的方式，通过广义的策略迭代 (第 4.6 节) 或由专家-批评家的方法来最大化预测。这样，一个代理就可以学会预测和控制大量的信号，而不仅仅是长期的奖励。为什么预测和控制信号比长期奖励有用呢? 这些都是辅助任务，因为它们是额外的、附加的，是最大化回报的主要任务。一个答案是，预测和控制多种多样的信号的能力可以构成一种强有力的环境模型。正如我们在第 8 章中看到的，一个好的模型可以使代理更有效地获得报酬。需要进一步的概念才能清楚地开发这个答案，因此我们将其推迟到下一节。首先，让我们考虑两种更简单的方法，在这两种方法中，大量不同的预测可以帮助增强学习代理。辅助任务可以帮助完成主任务的一种简单方法是，它们可能需要一些与主任务相同的表示。一些辅助任务可能更容易，更少的延迟和行动与结果之间更清晰的联系。如果在容易的辅助任务中可以及早发现好的特性，那么这些特性可能会显著地加快对主要任务的学习。没有必要的理由证明这是真的，但在很多情况下这似乎是合理的。例如，如果你学会了在短时间尺度上预测和控制你的传感器，比如秒，那么你可能会合理地提出物体的部分概念，这将极大地帮助预测和控制长期奖励。我们可以想象一个人工神经网络 (ANN)，其中最后一层被分割成多个部分，或者头部，每个人都在处理不同的任务。一个人可能为主要任务生成近似的值函数 (以奖励为累积量)，而另一个人可能为各种辅助任务生成解。所有的正面都可以通过随机梯度下降传播到同一个身体——网络的共享前一部分，然后在它的下至上一层试图形成表象，以支持所有正面。研究人员已经尝试了辅助任务，比如预测像素的变化，预测下一次的奖励，并预测回报的分布。在许多情况下，这种方法已经被证明可以极大地加速学习的主要任务 (Jaderberg 等人，2017)。类似地，多次提出多个预测作为指导国家估计的一种方法 (见第 17.3 节)。另一种学习辅助任务的简单方法是通过类比经典条件作用的心理现象 (第 14.2 节) 来最好地解释。理解经典条件作用的一种方式是，进化建立在一个反射 (非习

得的) 关联上，与一个特定信号的预测的特定行为相关联。例如，人类和其他许多动物似乎都有一种天生的反射，每当它们预测被戳到眼睛时，就会眨眼。这个预测是学来的，但是从预测到闭眼之间的联系是建立在这个基础上的，因此这个动物在它的眼睛里省下了许多没有保护的戳。同样，从恐惧到心率的增加，或者到冰冻的联系也可以建立起来。代理

设计师可以做一些类似的事情，通过设计 (不学习) 将特定事件的预测与预定的行为联系起来。例如，一辆自动驾驶汽车如果学会了预测未来是否会发生碰撞，那么当预测超过某个阈值时，就可以给它一个内置的反射，让它停下来，或者转开。或者考虑一个真空清洁机器人，它学会了在返回充电器之前预测它是否会耗尽电池电量，并且当预测变为非零的时候，它会本能地返回到充电器上。正确的预测将取决于房子的大小、机器人所在的房间以及电池的年龄，这些都是机器人设计者很难知道的。设计人员很难建立一个可靠的算法来决定是否返回到充电器的感官，但是从学习到的预测来看，这很容易做到。我们预见到许多可能的方法，如这种方法，学习的预测可以有效地结合内在的控制行为的算法。最后，也许辅助任务最重要的作用是超越我们在本书中所做的假设，即状态表示是固定的，并且给了代理。要解释这一作用，我们首先必须后退几步，以认识到这一假设的重要性以及消除它的意义。我们在第 17.3 节中这样做。

15.11 通过选项的时间抽象

MDP 形式主义的一个吸引人的方面是，它可以在许多不同的时间尺度上有效地应用于任务。一个人可以用它来正式地完成这样的任务: 决定用哪块肌肉去抓住一个物体，用哪架飞机去方便地到达一个遥远的城市，用哪份工作去过一个令人满意的生活。这些任务在它们的时间尺度上有很大的不同，但是每个任务都可以被有效地表述为 MDP，可以通过计划或学习过程来解决，如本书所述。所有这些都涉及到与世界的互动、顺序决策，以及一个有用的目标，即随着时间的推移积累回报，因此所有这些都可以写成 MDPs。虽然所有这些任务都可以用 MDPs 来表示，但是人们可能认为它们不能用一个 MDP 来表示。它们涉及如此不同的时间尺度，如此不同的选择和行动概念! 例如，在肌肉抽搐的水平上计划一次穿越大陆的飞行将是没有好处的。然而，对于其他任务，抓握、投掷飞镖或打棒球，低水平的肌肉抽搐可能正好合适。人们做所有这些事情都天衣无缝，似乎没有在不同的层次之间切换。MDP 框架可以同时扩展到所有级别吗? 也许可以。一种流行的想法是在一个详细的级别上，使用一个小的时间步骤，使 MDP 形式化，但是使用与许多基本的时间步骤相对应的扩展的行动过程，使计划能够在更高的级别上进行。要做到这一点，我们需要一个行动过程的概念，它可以扩展到多个时间步骤，并包含终止的概念。一般方法是政策制定这两个想法, , 与依赖政府的终止功能, , 如养狐业。我们把它们中的一对定义为一种被称为选项的广义行为概念。执行一个选项 = ? 吗? 在时间 t 是获取动作„ 从 (圣 • |), 然后在时间 t + 1 终止概率 (圣 + 1)。如果选择

不终止 t + 1, 然后从 选择 + 1(圣 • | + 1), 并选择终止在 t + 2 的概率 (圣 + 2), 等等, 直到最终终止。方便考虑低级操作 options-each 行动的特殊情况对应于一个选项? , 吗? 其政策选择行动 ((s)= a 年代), 其终止函数为零 (所有 年代 (s)= 0 +)。选项有效地扩展了动作空间。代理可以选择一个低级操作/选项，在一个时间步骤之后终止，或者选择一个扩展选项，在终止之前执行多个时间步骤。选项被设计成可以与低级操作互换。例如, 一个行为价值函数的概念 q 自然地推广到一个选项值函数, 它接受一个状态和选择作为输入, 并返回预期的返回状态, 从选择终止执行, 并且按照政策, 此后 。我们还可以将策略的概念概括为从选项而不是操作中选择的分层策略，在选择时，在选择时，执行到终止。有了这些想法，本书中的许多算法都可以推广到学习近似的选项-值函数和层次策略。在最简单的情况下，学习过程从选项启动“跳转”到选项终止，只有当选项终止时才会发生更新。更微妙的是，可以使用“内部选择”学习算法在每个时间步骤上进行更新，这通常需要策略外的学习。也许最重要的概括性观点是环境模型，如第 3 章，第 4 章和第 8 章所阐述的。行为的传统模型

是状态转移的概率和在每个状态下采取行为的预期立即奖励。传统的行为模型是如何推广到期权模型的? 对于选项，适当的模型是由两个部分组成的，一个对应于执行该选项导致的状态转换，另一个对应于沿途的预期累积回报。期权模型的奖励部分，类似于国家行动对 (3.5) 的预期回报，是。r(s,)。= e r1 + R2 + 2R3 + • • • + −1 r ?S0 = s A0: −1 , (17.2), 所有选项 和州 年代, 是随机时间步的选择根据 终止。注意整体折扣参数 的角色在这个 equation-discounting 根据 , 但根据 终止选项。选择模型的状态转换部分稍微有点微妙。模型的这一部分描述了每个可能的结果状态的概率 (如 (3.4))，但是现在这个状态可能会在不同的时间步长之后产生，每个时间步长都必须以不同的方式折现。 模型选项指定为每个州年代 可能开始执行, 和每个国家年代? 可能终止,

p(s ?| 年代,)。= ∞ 吗?k = 1 kPr Sk = s ? = k | S0 = 年代,A0:k−1 , 。(17.3)

注意, 因为 k 的因素, 这 p(s ?| 年代,) 不再是一个转移概率和不再总结 1 / s 的所有值吗?。(尽管如此，我们仍然使用 p 中的 | 符号)

上面对选项模型转换部分的定义允许我们制定 Bellman 方程和动态编程算法，这些算法适用于所有选项，包括作为特殊情况的低级操作。例如, 一般传达员状态方程的分层策略 的值

v =

Ω(s) (|)

r(年代,)+ ? s? p(s ?| 年代,)v (?)

, (17.4)

Ω(s) 表示一组选项的状态。如果 Ω(s) 只包括底层操作, 那么这个方程可以减少平时的贝尔曼方程 (3.14) 的一个版本, 当然除了 是包含在新的 p(17.3), 因此没有出现。同样, 也没有 相应的规划算法。例如，带有选项的值迭代算法，类似于 (4.10)

vk + 1(s)。= 最大 Ω(s)

r(年代,)+ ? s? p(s ?| 年代,)vk(?)

所有 年代。

如果 Ω(s) 包括所有可用的低级的行为在每个年代, 然后这个算法收敛到传统 v∗, 最优策略可以计算。然而, 它计划选项尤其有用, 当只有一个子集的可能的选择被认为是 (Ω(s)) 在每一个状态。值迭代将收敛到受限制的选项集的最佳层次策略。虽然这个策略可能不是最优的，但是收敛速度会快得多，因为考虑的选项较少，而且每个选项都可以跳过许多时间步骤。要计划有选择，必须给一个选择模型，或者学习它们。学习选项模型的一种自然方法是将其表示为 GVFs 的集合 (如前面部分所定义)，然后使用本书中介绍的方法学习 GVFs。不难看出，对于期权模型的奖励部分，如何做到这一点。一个仅仅选择一个养狐业的累积量奖励 (Ct = Rt), 其政策选择的政策 (=) 及其终止函数贴现率倍选择的终止功能 ((s)= • (s))。获得真正的细胞核然后等于奖励选择模型的一部分,v , ,C(s)= r(年代,), 和这本书中描述的学习方法可以用来近似。选项模型的状态转换部分只是稍微复杂一点。需要为选项可能终止的每个状态分配一个 GVF。我们不希望这些 GVFs 累积任何东西，除了在选项终止时，以及在终止处于适当状态时。这可以通过选择预测向状态 s 转变的 GVF 的累积量来实现。Ct = (St) • 圣 = s ?。选择 GVF 的策略和终止函数与选择模型的奖励部分相同。真正的 GVF 等于 s? 部分选项的状态转换关系模型、v ,C(s)= p(s ?| 年代, 再次), 这本书的方法可以用来学习。虽然这些步骤看起来都很自然，但是将它们组合在一起 (包括函数逼近和其他基本组件) 是非常具有挑战性的，并且超出了当前的技术水平。

练习 17.1 本节给出了折现情况的选项，但是当使用函数逼近时，折现可能不适用于控制 (第 10.4 节)。类似于 (17.4) 的层次策略的自然 Bellman 方程是什么，但对于平均奖励设置 (第 10.3 节)? 与 (17.2) 和 (17.3) 类似的选择模型的两个部分是什么? 吗?

15.12 观察和状态

在这本书中，我们将学到的近似值函数 (以及第 13 章中的策略) 作为环境状态的函数。这是第一部分中所介绍的方法的一个重大限制，其中所学习的值函数被实现为一个表，以便任何值函数都可以被精确地逼近; 这种情况相当于假设环境的状态完全被代理观察到。但在许多有趣的情况下，当然在所有自然智能的生命中，感官输入只提供有关世界状况的部分信息。有些物体可能被其他物体遮挡，或在代理后面，或在数英里之外。在这些情况下，潜在的环境状态的重要方面是不能直接观察到的，假设所学习的值函数是作为一个表在环境的状态空间中实现的，这是一个强烈的、不现实的、限制的假设。我们在第二部分中开发的参数函数逼近框架的限制要小得多，而且可以说，根本没有限制。在第 2 部分中，我们假设所学习的值函数 (和策略) 是环境状态的函数，但是允许这些函数被参数化任意地限制。函数近似值包含了部分可观测性的重要方面，这有点令人吃惊，并没有得到广泛的承认。例如，如果有一个状态变量不可观测，那么可以选择参数化，使近似值不依赖于该状态变量。其效果就像状态变量不可观测一样。因此，参数化情况下得到的所有结果都适用于不改变的局部可观测性。在这个意义上，参数化函数逼近的情况包括局部可观测的情况。然而，如果不更明确地对待局部可观测性，就无法研究许多问题。虽然我们不能在这里对它们进行全面的处理，但我们可以概述这样做所需的变化。有四个步骤。首先，我们要改变这个问题。环境不会发出它的状态，而只会发出观察信号——这些信号取决于它的状态，但就像机器人的传感器一样，只能提供有关它的部分信息。为了方便起见，在不丧失通用性的前提下，我们假定奖励是观察的直接的、已知的功能 (也许观察是一个矢量，而奖励是 is 的组成部分之一)。环境交互将没有显式状态或奖励, 但只会是一个交变序列 的操作和观察 Ot O: A0 O1 A1 O2 A2 O3 A3 O4 ⋯, 每一个结尾都有一个特殊的终端观察。

其次，我们可以从观察和行动的序列中恢复本书中使用的状态概念。让我们用“历史”这个词，加上符号“Ht”，来表示轨迹的初始部分，直到观察到:Ht。= A0, O1，⋯−1, 不。历史代表了我们可以知道的关于过去的大部分内容，而不需要查看数据流之外的内容 (因为历史是整个过去的数据流)。当然，历史会随着 t 而发展，会变得庞大而笨拙。国家的概念是对历史的一些简洁的总结，它和实际的历史预测未来一样有用。让我们弄清楚这到底意味着什么。作为历史的总结，国家必须是历史的函数，St = f(Ht)，并且要像预测整个历史一样对未来有用，它必须具有所谓的马尔可夫性质。形式上，这是函数 f 的性质。函数 f 有马尔可夫性质当且仅当 h 和 h 两个历史? 它被 f 映射到相同的状态 (f(h)=f(h?) 下一个观测的概率也相同，

(h)= f(h ?) 公关 Ot + 1 = o | Ht = h, 在 = = 公关 Ot + 1 = o | Ht = h ?= 一个,(17.5) o 和 如果 f 是马尔可夫, 然后圣 = f(Ht) 是一个国家用这个词在这本书。因此，我们把它称为马尔可夫状态，以区别于那些虽是历史总结但却没有马尔可夫性质的状态 (我们稍后将考虑)。马尔可夫状态是预测下一次观测的良好基础 (17.5)，但更重要的是，它也是预测或控制任何事物的良好基础。例如，让测试是未来可能发生的任何交替动作和观察的特定序列。例如, 一个三步测试 = a1o1a2 来标示,o2,a3,o3。给定特定历史 h 的这个测试的概率定义为

p(| h)。= 公关 Ot + 1 = o1,Ot + 2 = o2,Ot + 3 = o3 | Ht = h = a1,a2 + 1 =,+ 2 = a3 。(17.6)

如果 f 是马尔可夫 h? 任何两个历史映射到相同的状态下, 然后对任何测试 的长度, 其概率给定两个历史也必须是相同的:

(h)= f(h ?) p(| h)= p(| h ?)。(17.7)

换句话说，马尔可夫状态总结了历史上决定任何测试概率所需的所有信息。事实上, 它总结了所有必要的做任何预测, 包括任何养狐业, 表现最佳 (如果 f 是马尔可夫, 那么总有一个确定性的函数 这样选择。= (f(Ht)) 最优)。将强化学习扩展到局部可观测性的第三步是处理某些计算上的考虑。特别是，我们希望国家是历史的紧凑总结。例如，恒等函数完全满足马尔可夫 f 的条件，但仍然没有什么用处，因为相应的状态 St=Ht 会随着时间增长，变得难以处理，如前所述，但更根本的原因是它永远不会重现; 代理不会

遇到相同的状态两次 (在一个持续的任务中)，因此不能从表格学习方法中获益。我们希望

我们的状态和马尔可夫一样紧凑。关于如何获得和更新状态也有类似的问题。我们不想要一个包含整个历史的函数 f。相反，由于计算上的原因，我们更喜欢使用增量的、递归更新来获得与 f 相同的效果，该更新从 St 计算 St+1，包含下一个数据增量，At 和 Ot+1:

圣 + 1。= u(圣, 在 Ot + 1), 对所有 t 0, (17.8) 对于给定的第一状态 S0。函数 u 被称为状态更新函数。例如，如果 f 是恒等式 (St=Ht)，那么 u 仅仅通过在其上追加 At 和 Ot+1 来扩展 St。给定 f，总是有可能构造一个对应的 u，但它可能不方便计算，而且，就像恒等例子中那样，它可能不会生成一个紧态。状态更新函数是处理部分可观测性的任何代理体系结构的中心部分。它必须是有效的可计算的，因为在状态可用之前不能进行任何操作或预测。图 17.1 给出了这样一个代理体系结构的总体图。通过状态更新函数获得马尔可夫状态的一个例子是由流行的贝叶斯方法 (称为部分可观测的 MDPs) 提供的。在这种方法中，假定环境具有一个定义良好的潜伏状态 Xt，它作为环境观察的基础并产生环境观察，但是代理永远都不能使用它 (不要与代理用来进行预测和决策的状态 St 混淆)。自然的马可夫州，St，对于一个 POMDP 是在历史的潜在状态下的分布，称为信念状态。具体性, 假设通常的情况下, 有一个有限数量的隐藏状态,Xt 1,2,。d 。那么信念状态就是向量 St。= 圣 Rd 和组件

圣 [我]。= 公关 Xt = 我 | Ht , 所有可能的潜在的国家我 1,2,。d 。

信念状态保持相同的大小 (相同数量的组件)，但是不会增长。它还可以通过贝叶斯规则进行增量更新，假设一个人完全了解环境的内部工作。具体来说，belief-state 更新函数的第 i 个组件是

u(年代,o)[我]。= d ? x=1 s[x]p(i, o|x, a) d。x = 1 d ? x ?= 1 s[x]p(x ?o | x,) , (17.9)

阿, 对所有一个 啊, 和信念状态与组件 Rd 年代 [x],four-argument p 的函数通常不是一个 mdp(第三章), 但类似 POMDPs, 潜伏的状态:p(x ?o | x,)。x = 公关 Xt = ? 不 = o | Xt−1 = x,−1 = 。这种方法在理论工作中很流行，并且有很多重要的应用，但是它的假设和计算复杂度很差，我们不推荐它作为人工智能的一种方法。另一个 Markov 状态的例子是由预测状态表示，或 PSRs 提供的。PSRs 解决了 POMDP 方法的不足，即其代理状态 St 的语义基于环境状态 Xt，而这是从未被观察到的

一个

图 17.1: 概念代理体系结构，包括模型、计划器和状态更新函数。在这个案例中，世界接收行动 A 并发出观测结果。观察和行动的副本被国家更新函数 u 用来产生新的状态。新状态是策略和值函数的输入，生成下一个操作，也是输入对计划者 (和 u) 来说，最负责学习的信息流由虚线表示，虚线穿过它们所改变的方框。奖励 R 直接改变策略和价值函数。行为、奖励和状态改变了模型，它与计划者紧密合作，也改变了策略和价值功能。请注意, 规划器的操作可以与代理-环境交互解耦，而其他进程应该按照这个交互的锁步骤操作，以跟上到达的时间新数据。还要注意，模型和计划器并不直接处理观测，而是只处理观测与 u 生成的状态一起，可以作为模型学习的目标。

因此很难了解。在 PSRs 和相关的方法中，代理状态的语义是建立在对未来的观察和行为的预测上的，这是很容易观察到的。在 PSRs 中，马尔可夫状态定义为 d 特别选择的“核心”测试的概率的 d 向量 (17.6)。然后，通过状态更新函数 u 更新矢量，该函数类似于贝叶斯规则，但基于可观测数据的语义，这无疑使学习变得更容易。这种方法在许多方面都得到了扩展，包括末端测试、成分测试、强大的“光谱”方法，以及 TD 方法学习的闭环和时间抽象测试。一些最好的理论发展是关于可观察的操作员模型 (OOMs) 和顺序系统 (Thon, 2017)。在我们简要概述如何处理强化学习中的局部可观察性的第四个也是最后一个步骤是重新引入逼近。正如在第二部分的介绍中所讨论的，要雄心勃勃地接近人工智能，就必须接受近似。这对状态和值函数都是一样的。我们必须接受并使用一种近似的状态概念。近似状态在我们的算法中所起的作用和以前一样，所以我们继续使用记号 St 表示代理所使用的状态，尽管它可能不是 Markov。

也许关于近似状态最简单的例子就是最近的观测，St。= 不。当然，这种方法不能处理任何隐藏的状态信息。最好使用最后的 k 次观测和行动，St。= 不, 在 −−1。。。−k,k 1, 这可以通过状态更新功能, 只是变化的新数据和最古老的数据。这种 k 阶历史方法仍然非常简单，但

是与试图直接使用单个即时观察作为状态相比，可以极大地提高代理的能力。当马尔可夫特性 (17.5) 仅近似满足时会发生什么？不幸的是，当定义马尔可夫特性的一步预测变得更不准确时，长期预测性能会显著下降。较长期的测试、GVFs 和状态更新函数可能都不太接近。短期和长期的近似目标是不同的，目前没有有效的理论保证。尽管如此，仍然有理由认为本节概述的一般思想适用于近似情况。一般的想法是，对某些预测有利的状态对其他预测也有利(特别是，对一步预测足够的马尔可夫状态对所有其他预测也足够)。如果我们从马尔可夫案例的特定结果退回去，一般的想法与我们在第 17.1 节中讨论的多头脑学习和辅助任务相似。我们讨论了对辅助任务有利的表示方法如何也对主任务有利。综上所述，这些建议了一种对局部可观测性和表示学习的方法，在这种方法中，需要进行多个预测，并用于指导状态特征的构建。完全但不实用的马尔可夫性质所提供的保证被启发式取代，启发式认为，对某些预测有利的东西可能对其他预测有利。这种方法可以很好地利用计算资源。有了一台大型机器，人们就可以对大量的预测进行实验，可能更倾向于那些与最终感兴趣的或最容易可靠地学习的预测，或者根据其他一些标准。这里很重要的一点是，不要再手动选择预测了。代理应该这样做。这将需要一种预测的通用语言，以便代理能够系统地探索大量可能的预测，并从中筛选最有用的预测。特别是，POMDP 和 PSR 方法都可以应用于近似状态。状态的语义在生成状态更新函数时很有用，就像在这两种方法和 k 阶方法中一样。为了在状态中保留有用的信息，语义不需要是正确的。一些状态增强的方法，如 Echo state networks (Jaeger, 2002)，保留了关于历史的几乎任意的信息，但仍然可以很好地执行。有很多可能性，我们期望在这个领域有更多的工作和想法。学习近似状态的状态更新函数是强化学习中出现的表现学习问题的主要部分。

15.13 设计奖励信号

强化学习相对于监督学习的一个主要优势是强化学习不依赖于详细的教学信息: 产生奖励信号不依赖于知道行为者的正确行为是什么。但是，强化学习应用程序的成功与否，很大程度上取决于奖励信号对应用程序设计者的目标的框架程度，以及信号对实现目标进展的评估程度。基于这些原因，设计奖励信号是任何强化学习应用的关键部分。通过设计一个奖励的信号我们指的是设计一个代理环境的一部分, 负责计算每个标量奖励 Rt, 并将其发送给代理在 t。每次我们讨论术语第 14 章的末尾, 我们说 Rt 更像是一个信号生成在动物的大脑比像一个对象或事件在动物的外部环境。我们大脑中产生这些信号的部分经过了数百万年的进化，以很好地适应我们的祖先在努力向后代传播基因时所面临的挑战。因此，我们不应该认为设计一个好的奖励信号总是一件容易的事情! 一个挑战是设计一个奖励信号，这样当一个代理学习时，它的行为就会接近并最终达到应用程序设计者真正想要的。如果设计人员的目标简单且容易识别，比如找到一个定义明确的问题的解决方案，或者在定义明确的游戏中获得高分，那么这很容易实现。在这样的情况下，通常是根据代理在解决问题上的成功或改进分数的成功来奖励它。但有些问题涉及目标，这些目标很难转化为奖励信号。当问题需要代理熟练地执行复杂的任务或一组任务时，这一点尤为明显，比如需要一个有用的家用机器人助理。此外，强化学习代理可以发现意想不到的方法来让他们的环境产生回报，其中一些可能是不受欢迎的，甚至是危险的。对于任何基于优化的方法，如强化学习，这都是一个长期且关键的挑战。我们将在本书的最后部分 17.6 节中更多地讨论这个问题。即使有一个简单而容易识别的目标，稀疏奖励的问题也经常出现。频繁地提供不为零的奖励以使代理能够实现一次目标，更不用说学会如何在多个初始条件下有效地实现目标，这可能是一个艰巨的挑战。显然值得引发奖励的状态-行动对可能是很少的，而且在实现目标的过程中标记进展的奖励可能是很少的，因为进展是困难的，甚至是不可能被发现的。代理人可能会漫无目的地游荡很长一段时间 (明斯基 1961 年将其称为“高原问题”)。在实践中，设计奖励信号通常留给非正式的试错搜索，以产生可接受的结果。如果代理无法学习，学习太慢，或者学习错误的东西，那么设计师就会调整奖励信号，再试一次。为此，设计师根据他或她试图转化为奖励信号的标准来判

断代理人的表现，以便代理人的目标与他或她自己的目标匹配。如果学习太慢，设计者可能会尝试设计一个非稀疏的奖励信号，有效地指导整个 agent 与环境的交互过程中的学习。

通过奖励实现子目标的代理来解决稀疏的奖励问题是很诱人的，设计者认为这是实现总体目标的重要途径。但是，用善意的补充奖励来增加奖励信号，可能会导致代理的行为与预期的不同; 代理最终可能根本没有实现总体目标。提供这种指导的更好方法是不理会奖励信号，而是通过对其最终应该是什么的初步猜测来增强价值函数近似，或者通过对其特定部分的初步猜测来增强它。例如, 假设人愿意提供 v0:S→R 作为初始猜测真正的最优值函数 v∗„ 一个是使用线性函数近似特征 x:S→Rd。那么一个定义初始值函数逼近 v̂(s,w)。= w ？ x(s)+ v0(s), (17.10) 和往常一样更新权重 w。如果初始权值向量为 0，则初始值函数为 v0，而渐近解的质量则由特征向量决定。这种初始化可以对任意非线性逼近器和任意形式的 v0 进行，尽管不能保证总是能加速学习。针对稀疏奖励问题的一种特别有效的方法是心理学家 b • f • 斯金纳 (B. F. Skinner) 介绍的整形技术，并在第 14.3 节中进行了描述。这种技术的有效性依赖于这样一个事实: 稀疏奖励问题不仅仅是奖励信号的问题; 它们也存在于代理的策略中，以防止代理经常遇到奖励状态。塑造包括在学习过程中改变奖励信号，从一个并不稀疏的奖励信号开始，考虑到代理的初始行为，并逐渐将其修改为适合于原始兴趣问题的奖励信号。每一次修改都要进行，以便在给定代理当前行为的情况下经常对其进行奖励。代理面临一系列越来越困难的强化学习问题，在每个阶段学到的东西使得下一个更难的问题相对容易，因为代理现在遇到奖励的频率比以前没有遇到更容易的问题的经验时要高。这种塑形是训练动物的一种基本技术，在计算强化学习中也很有效。如果一个人不知道奖励应该是什么，但又有另一个人，也许是一个人，他已经是这个任务的专家，他的行为可以被观察到? 在这种情况下，我们可以使用不同的方法，如“模仿学习”、“示范学习”和“学徒学习”。“这里的想法是从专家代理中获益，但保留最终表现更好的可能性。”从专家的行为中学习可以通过直接学习监督学习或者利用所谓的“反向强化学习”提取奖励信号，然后使用带有奖励信号的强化学习算法来学习策略。Ng 和 Russell(2000) 研究的逆强化学习任务是尝试从专家的行为中恢复专家的奖励信号。这不能完全做到这一点，因为对于许多不同的奖励信号 (例如给予所有国家和行动相同的奖励的任何奖励信号)，政策是最优的，但也有可能找到合理的奖励信号。不幸的是，需要强有力的假设，包括对环境动力学和特征向量的了解

奖励信号是线性的。该方法还需要多次完全解决问题 (例如，通过动态编程方法)。尽管存在这些困难，Abbeel 和 Ng(2004) 认为逆强化学习方法有时可以比监督学习更有效地从专家的行为中获益。找到一个好的奖励信号的另一种方法是自动地尝试和错误地搜索我们上面提到的一个好的信号。从应用的角度来看，奖励信号是学习算法的一个参数。与其他算法参数一样，通过定义一个可行候选空间并应用优化算法，可以自动搜索一个好的奖励信号。优化算法通过运行带有该信号的强化学习系统，对每个候选奖励信号进行若干步的评估，然后通过一个“高级”目标函数对总体结果进行评分，该目标函数旨在编码设计者的真实目标，而忽略了 agent 的局限性。奖励信号甚至可以通过在线梯度上升得到改善，其中梯度是高级目标函数的梯度 (Sorg, Lewis, and Singh, 2010)。将这种方法与自然世界联系起来，优化高级目标函数的算法类似于进化，高级目标函数是由动物后代的数量决定的进化适应性。使用这种双层优化方法的计算实验——一种类似于进化，另一种类似于个体的强化学习——证实了仅仅凭直觉并不总是足以设计出好的奖励信号 (Singh, Lewis，和 Barto, 2009)。由高级目标函数评估的增强学习代理的性能可以非常敏感地关注代理的奖励信号的细节，这些细节是由代理的限制和它的行为和学习环境所决定的。这些实验也证明了 agent 的目标不应该总是与 agent 的设计者的目标一致。乍一看，这似乎有悖常理，但代理商可能不可能实现设计师的目标，无论其奖励信号是什么。代理必须在各种各样的约束条件下学习，比如有限的计算能力，有限的环境信息，或者有限的学习时间。当存在这样的约束时，学习实现与设计师目标不同的目标有时会比直接追求目标更接近设计师的目标 (Sorg, Singh, and Lewis, 2010;Sorg,2011)。在自然界中很容易找到这样的例子。因为我们不能直接评估大多数食物的营养价值，进化——我们的奖励信号的设计者——给了我们一个奖励信号，让我们去寻找特定的口味。虽然肯定不是绝对正确的 (确实，在某些方面与祖先环境不同的环境中可能有害)，但这弥补了我们的

许多局限性: 我们的感官能力有限，我们可以学习的时间有限，以及通过个人实验找到健康饮食的风险。同样，因为动物不能观察自己的进化适应性，所以客观的功能不能作为学习的奖励信号。相反，进化提供了对可观察到的进化适应性预测者敏感的奖励信号。最后，请记住，强化学习代理并不一定像一个完整的

生物体或机器人; 它可以是一个更大行为系统的组成部分。这意味着奖励信号可能会受到行为主体内部的东西的影响，比如动机状态，记忆，想法，甚至是幻觉。奖励信号也可能取决于学习过程本身的属性，比如衡量学习进展的程度。使奖励信号敏感信息等内部因素这些可以让一个代理学习如何控制它的“认知结构”, 以及掌握知识和技能, 很难从一个奖励的信号, 只取决于外部事件。类似这样的可能性导致了“内部激励强化学习”的概念，我们将在下一节的最后简要地讨论这个概念。

15.14 剩余问题

在这本书中，我们提出了人工智能强化学习方法的基础。粗略地说，这种方法是基于无模型和基于模型的方法共同工作的，如第 8 章的 Dyna 体系结构，结合第二部分中所开发的函数逼近。重点是在线和增量算法，我们认为它们甚至是基于模型的方法的基础，以及如何在非策略培训情况下应用这些算法。后者的全部基本原理仅在最后一章中提出。我们一直呈现 off-policy 学习作为一个吸引人的方式来处理探索/利用困境, 但只有在这一章我们讨论了同时学习许多不同的辅助任务养狐业和学习世界等级的 temporally-abstract 选择模型, 两者都涉及 off-policy 学习。正如我们在整本书中所指出的，还有许多有待解决的问题，本章讨论的更多研究方向也证明了这一点。但是假设我们很慷慨，对我们在书中所做的一切以及到目前为止在这一章中所概述的一切都给出了大致的轮廓。之后还剩下什么? 当然我们不能确定需要什么，但我们可以做一些猜测。在这一节中，我们强调了另外六个问题，在我们看来，这些问题仍然需要在未来的研究中加以解决。首先，我们仍然需要强大的参数函数逼近方法，这些方法在完全增量和在线设置中工作良好。基于深度学习和 ANNs 的方法是这一方向上的主要步骤，但是，仍然只能在大型数据集上进行批量训练，通过广泛的离线自我扮演进行训练，或者从多个代理在同一任务上的交叉经验中学习。这些设置和其他设置都是解决当今深度学习方法的基本限制的方法。深度学习方法很难在增量式的在线环境中快速学习，而这正是本书强调的强化学习算法最自然的地方。这个问题有时被描述为“灾难性干扰”或“相关数据”。“当新事物被发现时，它往往会取代以前学过的东西，而不是增加它，结果是旧的学习的益处消失了。”诸如“重播缓冲区”之类的技术通常用于保留和重播旧数据，以便其好处不会永久丧失。诚实的评价是，目前的深度学习方法不太适合在线学习。我们没有理由认为这种限制是不可克服的，而是认为是算法

这种方法，虽然同时保留了深度学习的优势，但还没有被设计出来。目前的大多数深度学习研究都是针对这一限制而不是消除它。第二 (也许是紧密相关的)，我们仍然需要学习特性的方法，以便后续学习能够很好地归纳。这个问题是一个普遍问题的实例，被称为“表示学习”，“建设性诱导”和“元学习”——我们如何使用经验，而不仅仅是学习一个给定的期望函数，而是学习归纳偏差，从而使未来的学习变得更好，从而更快? 这是一个古老的问题，可以追溯到上世纪五六十年代人工智能和模式识别的起源。这样的年龄应该会让人踌躇一下。也许没有解决办法。但同样有可能的是，找到解决办法并证明其有效性的时机尚未到来。如今，机器学习的规模比过去大得多，良好的表示学习方法的潜在好处也变得更加明显。我们注意到，自 2013 年以来，每年都有一个新的年度会议——国际学习代表大会 (International Conference on Learning representative)——探讨这一问题和相关话题。在强化学习环境中探索表现学习也不太常见。强化学习为这个旧问题带来了一些新的可能性，例如第 17.1 节讨论的辅助任务。在强化学习中，表征学习的问题可以通过学习第 17.3 节中讨论的状态更新函数来确定。第三，我们仍然需要可伸缩的方法来使用学习的环境模型进行规划。在 AlphaGo Zero 和计算机国际象棋等应用程序中，规划方法已经被证明是非常有效的，在这些应用程序

中，环境模型是根据游戏规则确定的，或者可以由人类设计师提供。但是，完全基于模型的强化学习 (环境模型从数据中学习，然后用于规划) 的情况非常少见。第 8 章中所描述的 Dyna 系统就是一个例子，但是正如这里所描述的，并且在后续的工作中，它使用了一个没有函数逼近的表格模型，这极大地限制了它的适用性。只有很少的研究包括学习过的线性模型，甚至更少的研究包括使用选项的时间抽象模型，如第 17.2 节所讨论的。在使用已学习的模型进行规划之前，需要做更多的工作。例如，学习模型需要有选择性，因为模型的范围会强烈地影响规划效率。如果一个模型关注最重要的选项的关键后果，那么规划可以是高效和快速的，但是如果一个模型包含不太可能被选择的选项的不重要后果的细节，那么规划可能几乎是无用的。环境模型的建立应考虑其状态和动态，以优化规划过程为目标。模型的各个部分应该被不断地监测，以达到它们对计划效率的贡献或减损程度。这个领域还没有解决这个复杂的问题，也没有设计考虑到它们的影响的模型学习方法。

有些人会说深度学习解决了这个问题，例如，第 16.5 节中描述的 DQN 说明了一个解决方案，但我们并不信服。目前还没有什么证据能证明这一点。学习单独解决代表学习问题是一种普遍而有效的方法。

在未来的研究中需要解决的第四个问题是自动选择代理工作的任务，并使用这些任务来构建其开发能力。在机器学习中，设计人员通常会设置学习代理希望掌握的任务。因为这些任务是预先知道的，并且是固定的，所以可以将它们构建到学习算法代码中。但是，展望未来，我们希望代理能够对它应该尝试掌握的任务做出自己的选择。这些可能是已经知道的特定的总体任务的子任务，或者它们可能是为了创建构建块，从而允许更高效地学习代理在将来可能面临的许多不同任务，但是目前还不知道。这些任务可能类似于第 17.1 节中讨论的辅助任务或 GVFs，或者按照第 17.2 节讨论的选项解决的任务。例如，在形成 GVF 时，累积量、策略和终止函数应该是什么？当前的艺术状态是手动选择这些任务，但更大的权力和一般性将来自于自动地做出这些任务选择，特别是当它们来自于代理先前构建的由表示学习或以前的子问题的经验所构建的结果时。如果 GVF 设计是自动化的，那么设计选择本身就必须显式地表示出来。与其让设计人员考虑并将任务选择嵌入到代码中，还不如让它们在机器本身中进行设置、更改、监视、过滤和自动搜索。然后，任务可以分层地构建在其他任务上，就像 ANN 中的特性一样。任务就是问题，ANN 的内容就是这些问题的答案。我们期望有一个完整的问题层次来匹配现代深度学习方法提供的答案层次。我们想强调的第五个问题是行为和学习之间的相互作用通过好奇心的计算模拟。在这一章中，我们设想了一个场景，在这个场景中，许多任务同时被学习，使用非策略方法，从相同的经验流中学习。所采取的行动当然会影响这一经验流，而这反过来又将决定发生了多少学习，以及学习了哪些任务。当奖励不存在，或者不受行为的强烈影响时，代理可以自由地选择在某种意义上最大限度地学习任务的行为，也就是说，使用学习进度的某种度量作为内部或“内在”的奖励，实现好奇心的计算形式。除了测量学习进度外，内在奖励除了其他可能性外，还可以表示收到了意外的、新奇的或其他有趣的输入，或者可以评估代理在其环境中引起变化的能力。在这些方法中产生的内在奖励信号可以由一个代理通过定义辅助任务、GVFs 或选项来为自己设置任务，如上面所讨论的，这样通过这种方式学习的技能可以帮助代理者掌握未来任务的能力。结果是一种类似游戏的计算模拟。对这种内在奖励信号的使用进行了许多初步研究，在这一一般领域仍有令人兴奋的未来研究课题。在未来的研究中需要注意的最后一个问题是开发方法，使在物理环境中嵌入增强学习因子是可以接受的安全的。这是未来研究最紧迫的领域之一，我们将在下一节中进一步讨论。

15.15 人工智能的未来

上世纪 90 年代中期，当我们正在撰写这本书的第一版时，人工智能正在取得重大进展，并对社会产生了影响，尽管人工智能的前景仍是推动发展的主要因素。机器学习是这一观点的一部分，但它尚未成为人工智能不可或缺的一部分。到今天，这一承诺已经转变为正在改变千

百万人生活的应用程序，而机器学习作为一项关键技术已经有了自己的意义。当我们写第二版时，人工智能中一些最显著的发展已经涉及到强化学习，最著名的是“深层强化学习”——用深层人工神经网络的函数逼近来强化学习。我们正处于一波人工智能在现实世界中的应用浪潮的开端，其中许多应用将包括强化学习，深入地，否则，将以难以预测的方式影响我们的生活。但是，大量成功的实际应用并不意味着真正的人工智能已经到来。尽管在许多领域取得了巨大的进步，人工智能与人类甚至其他动物的智能之间的鸿沟仍然巨大。在某些领域，甚至是像 Go 这样令人生畏的领域，都可以实现超人的性能，但开发像我们这样的系统仍然是一个巨大的挑战，这些系统是完整的、具有一般适应性和解决问题的技能、情感复杂性、创造力以及从经验中快速学习的能力。通过与动态环境的交互作用，强化学习，随着它在未来的发展，将成为具有这些能力的代理的一个重要组成部分。强化学习与心理学和神经科学的联系 (第 14 章和第 15 章) 强调了它与人工智能的另一个长期目标的相关性: 阐明关于大脑的基本问题以及它是如何从大脑中浮现出来的。强化学习理论已经有助于我们理解大脑的奖励、动机和决策过程，我们有充分的理由相信，通过它与计算精神病学的联系，强化学习理论将有助于治疗包括药物滥用和成瘾在内的精神障碍的方法。强化学习对未来的另一个贡献是对人类决策的帮助。通过在模拟环境中强化学习而产生的政策，可以在教育、医疗、交通、能源和公共部门资源分配等领域为人类决策者提供建议。特别相关的是强化学习的关键特征，它考虑到决策的长期后果。这在《西洋双陆棋》和《Go》这类游戏中是非常明显的，在这些游戏中，强化学习的一些最令人印象深刻的结果已经被证明，但它也是影响我们的生活和我们的星球的许多高风险决策的属性。强化学习遵循指导人类决策的相关方法，这些方法在过去由许多学科的决策分析师开发。利用先进的函数逼近方法和大量的计算能力，增强学习方法有可能克服将传统的决策支持方法扩展到更大、更复杂的问题上的一些困难。

人工智能的快速发展已导致警告，人工智能对我们的社会，甚至对人类本身构成严重威胁。著名科学家和人工智能先驱 Herbert Simon 在 2000 年 CMU 的陶器研讨会上的演讲中预见到了我们今天听到的警告 (Simon, 2000)。他永恒的承诺和危险之间的冲突的任何新知识, 提醒我们普罗米修斯, 希腊神话的现代科学的英雄, 谁偷了来自上帝的火, 造福人类, 和潘多拉的盒子可以打开一个小和无辜的行动释放世界上数不清的危险。尽管西蒙承认这种冲突是不可避免的，但他敦促我们认识到，作为未来的设计者，而不仅仅是旁观者，我们所做的决定会使规模向普罗米修斯的方向倾斜。强化学习确实如此，它可以造福社会，但如果不小心使用，也可能产生不良后果。因此，涉及强化学习的人工智能应用的安全性是一个值得关注的课题。强化学习代理可以通过与现实世界或者与现实世界的某个部分进行交互，或者通过这两个经验来源的混合来学习。模拟器提供了一个安全的环境，在这个环境中，一个代理可以探索和学习，而不会对自身或环境造成真正的损害。在大多数当前应用程序中，策略是从模拟经验中学习的，而不是直接与真实世界交互。除了避免不良的实际后果, 从模拟学习经验可以为学习提供无限的数据, 通常比需要以较低的成本获得实际经验, 因为比实时模拟通常运行更快, 更快地学习经常会出现比它依赖于真实的体验。然而，强化学习的全部潜力需要强化学习媒介嵌入到真实世界的体验流中，在那里它们在我们的世界中行动、探索和学习，而不仅仅是在它们的世界中。毕竟，强化学习算法——至少我们在本书中关注的那些算法——是为了在网上学习而设计的，它们模仿了动物如何在非静止和敌对环境中生存的许多方面。在现实世界中嵌入强化学习因子可以实现人工智能增强和扩展人类能力的承诺。主要原因希望强化学习代理行为和学习在现实世界中是它常常是困难的, 有时是不可能的, 来模拟现实世界的经验和足够的忠诚产生的政策, 是否得到了强化学习或其他方法, 工作的好时候安全指导实际行动。这尤其适用于那些动态依赖于人类行为的环境，如教育、医疗、交通和公共政策领域，这些领域肯定可以从改进的决策制定中获益。然而，真正需要注意的是，对于现实世界中的嵌入式代理，关于人工智能潜在危险的警告。其中一些警告与强化学习特别相关。因为强化学习是基于优化的，它继承了所有优化方法的优缺点。缺点是设计目标函数，或者在强化学习中设计奖励信号，这样优化就产生了

避免不希望的结果。我们在第 17.4 节中说过，强化学习代理可以发现意想不到的方法来让他们的环境产生回报，其中一些可能是不受欢迎的，甚至是危险的。当我们指定我们希望

一个系统只能间接学习的内容时，就像我们在设计一个强化学习系统的奖励信号时一样，我们将不知道在它的学习完成之前，代理将如何完成我们的愿望。这并不是强化学习的新问题;它在文学和工程学上都有着悠久的历史。例如，在歌德的诗歌《魔法师的学徒》(歌德，1878)中，《学徒》用魔法使一把扫帚完成了取水的工作，但由于学徒对魔法知识的不充分，导致了意想不到的洪水。在工程背景下，控制论的创始人诺伯特 • 维纳 (Norbert Wiener) 半个多世纪前通过讲述“猴爪”的超自然故事 (维纳，1964) 来警告这个问题:“……”它赋予你所要求的，而不是你应该要求的或你想要的”(第 59 页)。尼克 • 博斯特罗姆 (Nick Bostrom)(2014) 在《现代语境》(modern context) 中也详细讨论了这个问题。任何有强化学习经验的人都可能看到他们的系统发现了意想不到的方法来获得大量奖励。有时意想不到的行为是好的: 它以一种新的方式解决问题。在其他情况下，代理学习的内容违反了系统设计者可能从未考虑过的考虑。如果一个代理人在现实世界中采取行动，而没有机会对其行为进行审查，或者很容易打断其行为，那么仔细设计奖励信号是必不可少的。尽管可能会产生意想不到的负面影响，优化已经被工程师、架构师和其他设计对世界产生积极影响的人使用了数百年。由于优化方法的应用，我们在我们的环境中有很多优点。为了降低优化的风险，已经开发了许多方法，如添加硬约束和软约束、将优化限制为鲁棒和风险敏感的策略，以及使用多个目标函数进行优化。其中一些方法已经被用于强化学习，需要更多的研究来解决这些问题。确保增强学习代理的目标与我们自己的目标一致仍然是一个挑战。另一个挑战是，强化学习主体在现实世界中的行为和学习不仅仅是关于他们最终可能学到的东西，而是关于他们在学习时的行为。如何确保代理获得足够的经验来学习高性能策略，同时不损害其环境、其他代理或其自身 (或者更实际地说，同时将损害的可能性保持在可接受的低水平)? 这个问题对强化学习也不是新奇的或独特的。嵌入式强化学习的风险管理和缓解类似于控制工程师从使用自动控制开始就必须面对的情况，在这种情况下，控制器的行为可能会产生不可接受的、可能是灾难性的后果，如控制飞机或精细的化学过程。控制应用程序依赖于仔细的系统建模、模型验证和广泛的测试，并且有一个高度发达的理论体系，旨在确保在控制系统的动态不完全被控制的情况下，为使用而设计的自适应控制器的收敛性和稳定性。理论保证从来都不是铁板钉钉的，因为它们依赖于数学基础上的假设的有效性，但如果没有这个理论，再加上风险管理和缓解实践，自动控制——自适应和其他控制——就不会是这样

在改进我们所依赖的过程的质量、效率和成本效益方面，它今天是有益的。未来强化学习研究最紧迫的领域之一是采用和扩展控制工程中开发的方法，使强化学习剂充分嵌入物理环境中成为可接受的安全方法。最后，我们回到西蒙的呼吁，让我们认识到我们是未来的设计师，而不仅仅是观众。通过我们作为个人所做的决定，以及我们对社会治理方式施加的影响，我们可以努力确保一项新技术可能带来的好处超过它可能带来的危害。在加强学习的情况下，有充分的机会可以做到这一点，这有助于提高我们星球上的生活质量、公平和可持续性，但这也能释放新的危险。目前的威胁是人工智能应用所造成的工作岗位的流失。仍然有充分的理由相信人工智能的好处可以超过它所造成的破坏。在安全性方面，强化学习可能产生的危害与在优化控制方法的相关应用中成功管理的危害并没有完全不同。随着增强学习在未来的应用程序中逐渐进入现实世界，开发人员有义务遵循为类似技术进化的最佳实践，同时扩展它们以确保 Prometheus 占上风。

15.16 书目的和历史的言论

17.1 一般价值函数首先由 Sutton 和同事明确识别 (萨顿,1995; 萨顿 et al .,2011; 莫达耶 (Modayil, White and Sutton, 2013)。Ring (in preparation) 用 GVFs(“预测”) 开发了一项广泛的思想实验，尽管还没有发表出来，但这一实验已经很有影响力了。Jaderberg 等人 (2017) 首次演示了增强学习中的多脑学习。Bellemare、Dabney 和 Munos(2017) 的研究表明，预测更多关于奖励分配的事情可以显著地加速学习以优化预期，这是辅助任务的一个例子。此后，许多人开始从事这方面的研究。经典条件作用的一般理论是学习过的预测，以及对预测的内

在的、反射性的反应，这在心理学文献中并没有得到明确的阐述。Modayil 和 Sutton(2014) 将其描述为机器人和其他代理的工程方法，称其为“巴甫洛夫控制”，暗指其根源于古典条件作用。

17.2 作为选择的临时性抽象的行动过程的形式化是内含的由 Sutton、Precup 和 Singh(1999) 指导，以 Parr(1998) 和 Sutton(1995) 的前期工作为基础，以及关于半千年发展目标的经典工作 (例如，参见 Puterman, 1994)。Precup(2000) 博士论文充分发展了期权的思想。这些早期工作的一个重要限制是，它们没有使用函数逼近来处理偏离政策的情况。一般情况下，期权内学习需要策略外学习，这在当时无法用函数近似可靠地完成。虽然现在我们使用函数近似有很多稳定的非政策学习方法，但在这本书出版的时候，它们与期权思想的结合并没有得到显著的探索。Barto 和 Mahadevan(2003) 和亨斯特 (2012) 回顾了时间抽象的选项形式主义和其他方法。使用 GVFs 实现选项模型之前没有描述过。我们的报告使用 Modayil、White 和 Sutton(2014) 引入的技巧来预测政策终止时的信号。在学习函数近似的选项模型的少数作品中，有 Sorg 和 Singh(2010)、Bacon、Harb 和 Precup(2017)。在文献中，选择和期权模型到平均奖励设置的扩展还没有被开发出来。

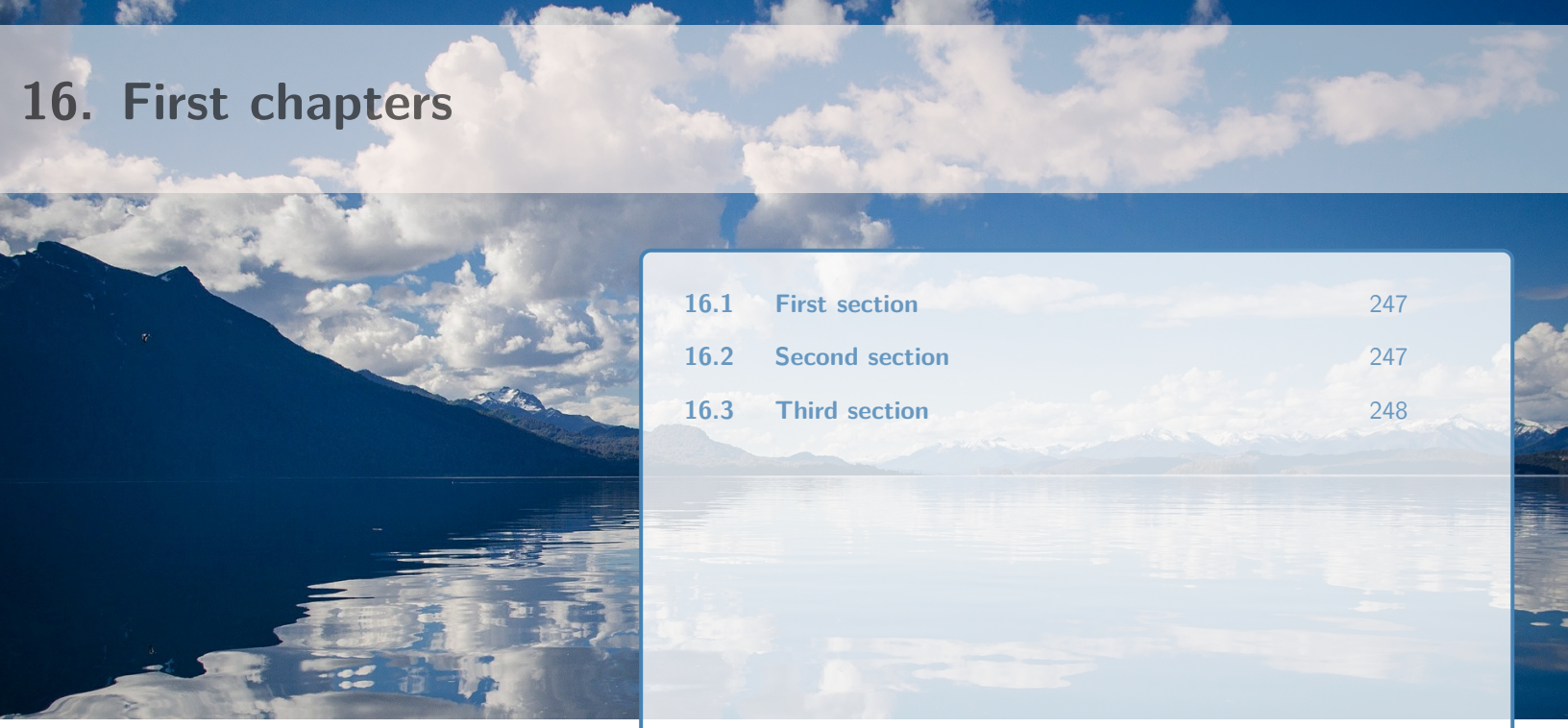
17.3 Monahan(1982) 很好地介绍了 POMDP 方法。psr 实验由 Littman, Sutton 和 Singh(2002) 提出。OOMs 是由 Jaeger(1997, 1998, 2000) 提出的。序列系统统一了 PSRs、oom 和其他许多工作，在 Michael Thon 的博士论文 (2017) 中被引入; 索恩和 Jaeger,2015)。时间关系网络的扩展。

由 Tanner(2006) 开发;Sutton 和 Tanner, 2005) 然后扩展到选项 (Sutton, Rafols，和 Koop, 2006)。用非马尔可夫状态表示的强化学习理论是由辛格、贾可拉和约旦明确提出的 (1994 年);Jaakkola, Singh，和 Jordan, 1995)。Chrisman(1992)、McCallum(1993, 1995)、Parr 和 Russell(1995)、Littman、Cassandra 和 Kaelbling(1995) 以及 Lin 和 Mitchell(1992) 提出了早期的增强学习方法。

17.4 在强化学习中包含建议和教学的早期努力包括 Lin (1992)，Maclin 和 Shavlik (1994)，Clouse (1996)，Clouse 和 Utgoff(1992)。Skinner 的整形不应该与 Ng、Harada 和 Russell(1999) 引入的“基于潜力的整形”技术混淆。他们的技术已经被 Wiewiora(2003) 证明等同于提供值函数的初始逼近的简单思想，如 (17.10)。

17.5 我们推荐 Goodfellow, Bengio 和 Courville(2016) 的《铁饼》今天的深度学习技巧。McCloskey 和 Cohen(1989)、Ratcliff(1990) 和 French(1999) 提出了对 ANNs 进行灾难性干预的问题。重播缓冲区的概念是 Lin(1992) 提出的，并在 Atari 游戏系统的深度学习中得到了突出的应用 (第 16.5 节，Mnih 等，2013,2015)。明斯基 (1961) 是最早发现表征学习问题的学者之一。在少数几个考虑计划学习的作品中，大约有 Kuvayev 和 Sutton(1996)、Sutton、Szepesvari、Geramifard、Bowling(2008)、Nouri 和 Littman(2009)、Hester 和 Stone(2012)。在人工智能中，有必要在模型构建中进行选择，以避免规划放缓。一些经典的作品是由明顿 (1990)、塔姆贝 (Tambe)、纽维尔 (Newell) 和罗森布鲁姆 (Rosenbloom)(1990) 合著的。Hauskrecht、Meuleau、Kaelbling、Dean 和 Boutilier(1998) 在具有确定性选项的 MDPs 中显示了这种影响。施米德胡贝尔 (1991a, b) 提出，如果奖励信号是一个代理环境模型改善速度的函数，那么好奇号之类的东西会产生什么结果。Klyubin、Polani 和 Nehaniv(2005) 提出的赋权函数是一种信息论上的对代理人控制环境能力的度量，这种能力可以作为一种内在的奖励信号。Baldassarre 和 Mirolli(2013) 是研究人员从生物学和计算角度研究内在奖励和动机的一组贡献，包括对“内在激励的强化学习”的观点，用 Singh、Barto 和 Chentenez(2004) 引入的术语。参见 Oudeyer 和 Kaplan(2007)、Oudeyer、Kaplan 和 Hafner(2007) 和 Barto(2013)。

16. First chapters



This first chapter illustrates how to use various elements of this text book template, such as definitions, theorems and exercises. You may want to start each chapter with a meta summary like this one, to explain to the reader what the chapter is all about, why it is important and how it fits into the bigger picture of the book. Another useful tip is to put the contents of each chapter into a separate LaTeX file and then use the command `\input{}` *to include the chapter in the main document.*

16.1 First section

Let's start out with the following theorem.

Theorem 16.1 (Logic algebra) Let P, Q and R be logical propositions (true or false). Then the following propositions are true:

$$
\begin{array}{lll}
P \wedge Q \Leftrightarrow Q \wedge P & P \vee Q \Leftrightarrow Q \vee P & \text{(commutative laws)} \\
(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R) & (P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R) & \text{(associative laws)} \\
P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R) & P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R) & \text{(distributive laws)} \\
\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q & \neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q & \text{(De Morgan's laws)}
\end{array}
$$

Proof. We prove the first of De Morgan's laws and leave the proofs of the remaining propositions as exercises. To prove the statement, we create a truth table and fill in all possible values (true or false) for the propositions P and Q. Each of these propositions can be either true or false and we thus obtain the following truth table with four cases:

It follows that the statement we want to prove (the equivalence $\Leftrightarrow$) is always true (a *tautology*), which proves the statement. □

16.2 Second section

We begin our next section with the following central definition.

Definition 16.1 (Rational Cauchy sequence) A rational Cauchy sequence is a rational sequence $(x_n)_{n=0}^{\infty}$ such that

$$\forall \epsilon \in \mathbb{Q}_+ \, \exists N \in \mathbb{N} : m, n \geq N \Rightarrow |x_m - x_n| < \epsilon. \tag{16.1}$$

In other words, for each (small) rational number $\epsilon > 0$ there is a (big) number N such that the distance $|x_m - x_n|$ between x_m and x_n is less than ϵ if both m and n are larger than or equal to N.

A remark may be in order here. This definition is concerned with rational *Cauchy sequences. We will later encounter a similar definition of* real *Cauchy sequences.*

Exempel 16.1 (Solving the equation $x^2 = 2$) Consider the equation $x^2 = 2$. It is easy to prove that this equation does not have any rational solutions. However, consider the following iteration formula:

$$x_n = \frac{x_{n-1} + 2/x_{n-1}}{2}, \tag{16.2}$$

where $n = 1, 2, 3, \ldots$ and $x_0 = 1$. The resulting sequence of rational numbers quickly approaches a number in the vicinity of $x = 1.4142135623731$:

$$\begin{aligned}
x_0 &= 1 \\
x_1 &= (x_0 + 2/x_0)/2 &&= 1.5 \\
x_2 &= (x_1 + 2/x_1)/2 &&\approx 1.4166666666667 \\
x_3 &= (x_2 + 2/x_2)/2 &&\approx 1.41421568627\,45 \\
x_4 &= (x_3 + 2/x_3)/2 &&\approx 1.41421356237\,47 \\
x_5 &= (x_4 + 2/x_4)/2 &&\approx 1.41421356237\,31 \\
x_6 &= (x_5 + 2/x_5)/2 &&\approx 1.41421356237\,31 \\
x_7 &= (x_6 + 2/x_6)/2 &&\approx 1.41421356237\,31 \\
x_8 &= (x_7 + 2/x_7)/2 &&\approx 1.41421356237\,31 \\
x_9 &= (x_8 + 2/x_8)/2 &&\approx 1.41421356237\,31 \\
x_{10} &= (x_9 + 2/x_9)/2 &&\approx 1.41421356237\,31
\end{aligned}$$

We will later see that this iteration, or any other equivalent iteration, defines the real number $\sqrt{2}$.

16.3 Third section

Now let's move on to the definition of the real number system. This may be defined in a multitude of ways, one of which is to think about a real number as a rational Cauchy sequence, or rather the equivalence class of Cauchy sequences "converging to" that number.

Definition 16.2 (The real numbers $\mathbb{R}$) The real numbers $\mathbb{R}$ is the set of all equivalence classes of rational Cauchy sequences.

Now that this is settled, lets prove the completeness of the real number system.

Theorem 16.2 (The completeness of the real numbers) Let $(x_n)_{n=0}^{\infty}$ be a sequence of real numbers. Then $(x_n)_{n=0}^{\infty}$ is convergent if and only if it is also a real Cauchy sequence.

Proof. Write $x_m = [(x_{mn})_{n=0}^{\infty}]$ where x_{mn} is the nth number in a rational Cauchy sequence representing the real number x_m. And so on.... □

For further reading, there are several excellent works that one could cite, such as [**Tao2006**, **Turing1936**].

Exercises

Exercise 16.1 Let $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$. Determine the following sets.
(a) $A \cup B$ (b) $A \cap B$ (c) $A \setminus B$ (d) $A \times B$

Exercise 16.2 Let $A = \{1, 3, 5, 7, 9\}$ and $B = \{2, 4, 6, 8, 10\}$. Determine the following sets.
(a) $A \cup B$ (b) $A \cap B$ (c) $A \setminus B$ (d) $A \times B$

Exercise 16.3 Let $A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$ and $C = \{3, 4, 5\}$. Determine the following sets.
(a) $A \cup B \cup C$ (b) $A \cap B \cap C$ (c) $(B \setminus A) \cap C$ (d) $(A \times B) \times C$

Problem

Problem 16.1 Interpret the following set definition (Russell's paradox) and discuss whether $X \in X$ or $X \notin X$:

$$X = \{x \mid x \notin x\}. \tag{16.3}$$

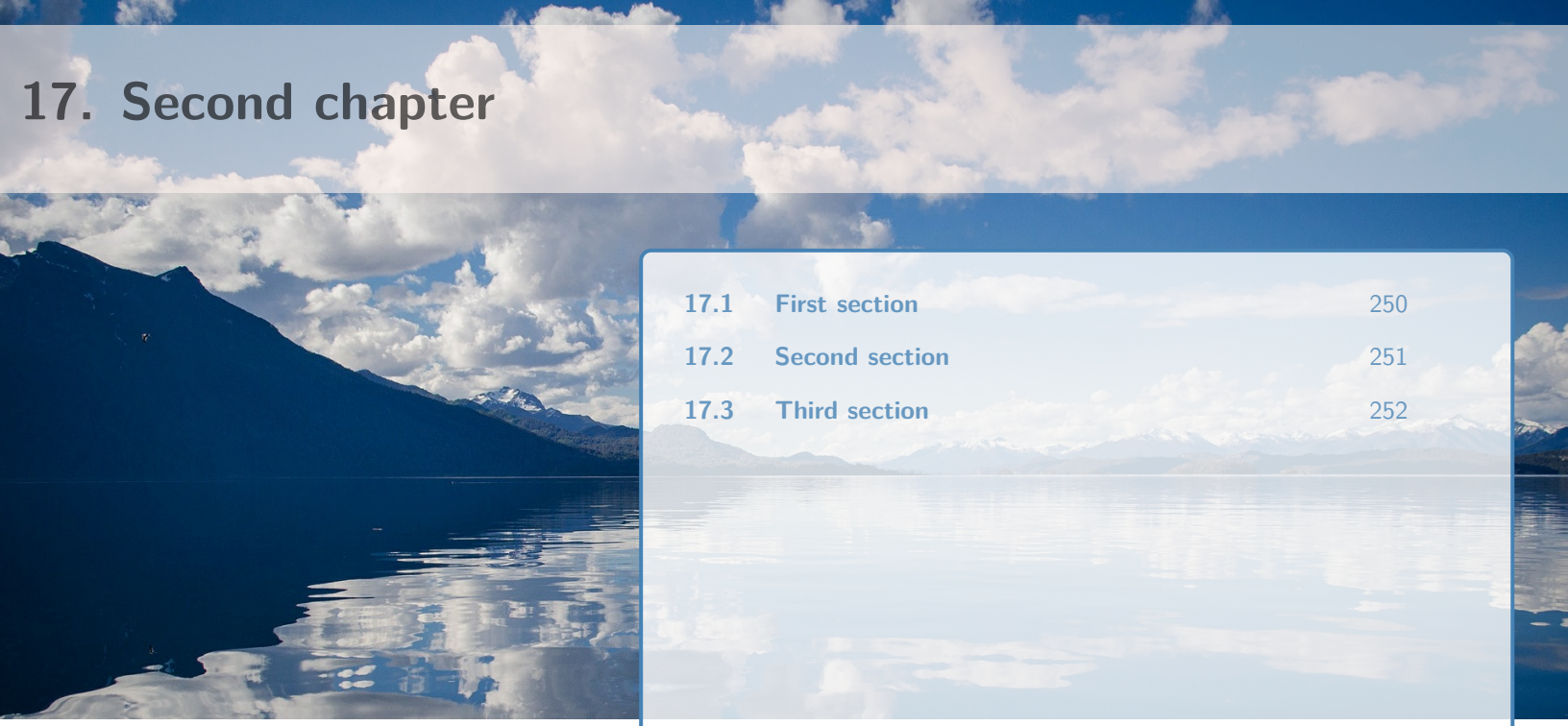
Computer exercises

Computer exercise 16.1 Write a program that generates the sequence $(x_n)_{n=0}^{100}$ for $x_n = n$.

Computer exercise 16.2 Write a program that generates the odd numbers between 1 and 100.

Computer exercise 16.3 Write a program that computes the sum $\sum_{n=0}^{100} x_n$ for $x_n = n$.

17. Second chapter



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

17.1 First section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem.

Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

17.2 Second section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

17.3 Third section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc

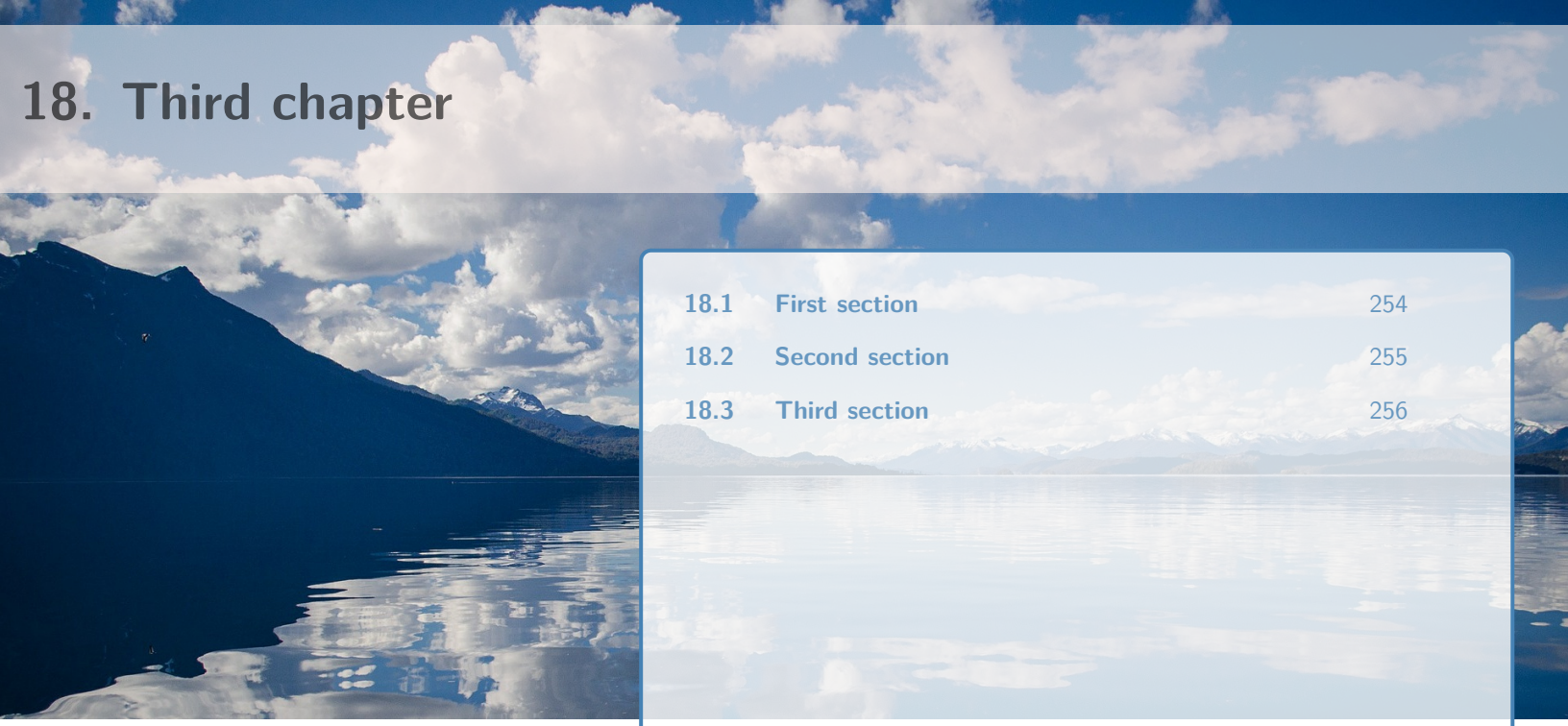
quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

18. Third chapter



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

18.1 First section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem.

Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

18.2 Second section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

18.3 Third section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc

quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.