动态规划与强化学习 (样章)[1]

德梅萃 · P. 博赛卡斯（Dimitri P. Bertsekas）著
李宇超（Yuchao Li）译

2025 年 10 月 17 日

[1]欢迎发送建议和评论到yuchaotaigu@gmail.com

版权与译者声明

关于作者

Dimitri Bertsekas 曾在希腊国立雅典技术大学学习机械与电气工程，之后从麻省理工学院获得系统科学博士学位。他曾先后在斯坦福大学工程与经济系统系和伊利诺伊大学香槟分校的电气工程系任教。1979 年以来，他一直在麻省理工学院电机工程与计算机科学系任教，现担任麦卡菲工程教授。2019 年，他加入亚利桑那州立大学计算、信息与决策工程学院并担任富尔顿教授。

Bertsekas 教授的研究涉及多个领域，包括确定性优化、动态规划、随机控制、大规模与分布式计算以及数据通信网络。他已撰写 19 本著作以及众多论文，其中数本著作在麻省理工学院被用作教材，包括《动态规划与最优控制》、《数据网络》、《概率导论》、《凸优化算法》以及《非线性规划》。

Bertsekas 教授因其著作《神经元动态规划》（与 John Tsitsiklis 合著）荣获 1997 年 INFORMS 授予的运筹学与计算机科学交叉领域的杰出研究成果奖、2001 年美国控制协会 John R. Ragazzini 奖以及 2009 年 INFORMS 说明写作奖、2014 年美国控制协会贝尔曼遗产奖、2014 年 INFORMS 优化学会 Khachiyan 终身成就奖、2015 年 MOS/SIAM 的 George B. Dantzig 奖、2018 年 INFORMS 的冯诺依曼理论奖，以及 2022 年 IEEE 控制系统奖。2001 年，他因为“基础性研究、实践并教育优化/控制理论，特别是在数据通信网络中的应用”当选美国工程院院士。

目录

第一章　动态规划与强化学习方法

本章有多个目的：

(a) 概述精确动态规划（英文为 *dynamic programming*，简写作 *DP*）方法，重点放在次优解法上。*(To provide an overview of the exact dynamic programming methodology, with a focus on suboptimal solution methods.)* 我们将首先讨论有限阶段问题，这类问题涉及一系列有限的连续决策，因此在概念和分析上都更为简单。我们将分别探讨确定性有限阶段问题（第 1.2 节）和随机性有限阶段问题（第 1.3 节）。其原因是确定性问题相对简单且具有一些有利特性，能够应用更多种类的方法求解。值得注意的是，这类问题包含了具有挑战性的离散优化和组合优化问题，而这些问题可以通过本书重点探讨的部分强化学习（英文为 reinforcement learning，简写作 RL）方法得到有效解决。我们还将简要讨论更为复杂的无穷阶段方法（第 1.4 节），，并在附录 A 中进行更为详细的阐述。关于更完整的论述可参考作者的动态规划教材 [Ber12]、[Ber17]，强化学习著作 [Ber19b]、[Ber20b]，以及《神经动态规划》专著 [BT96]。

(b) 概括主要的强化学习方法，重点聚焦于基于值空间的近似方法。*(To summarize the principal RL methodologies, with primary emphasis on approximation in value space.)* 这一基于近似动态规划的框架，正是阿尔法零、阿尔法围棋（AlphaGo）、TD-Gammon 等相关程序的核心，同时也是模型预测控制（英文为 model predictive control，简写作 MPC）方法的基础——模型预测控制是控制系统设计的主要方法之一。我们将在后续章节进一步阐述，基于值空间的近似方法为应用强化学习技术解决离散优化和整数规划问题提供了切入点。

(c) 阐述值空间近似的主要原则，并将其划分为离线训练与在线执行算法两部分。（*To explain the major principles of approximation in value space, and its division into the off-line training and the on-line play algorithms.*）此处的一个核心观点在于，这两个算法通过牛顿法的计算方法（用于求解问题的贝尔曼方程）相互关联。这一视角近期在笔者的《策略前展、策略迭代……》[Ber20b] 一书及以图示为主的《阿尔法零对……的启示》[Ber22b] 专著中得以发展，贯穿整个课程，并在第 1.5 节针对直观且重要的线性二次型问题类别进行了讨论。

本章还将有选择性地讨论近似动态规划和强化学习中的一些核心算法，包括策略前展与策略迭代。关于动态规划/强化学习的更广泛探讨可参阅强化学习著作 [Ber19b]、[Ber20b]，动态规划教材 [Ber12]、[Ber17]，神经元动态规划专著 [BT96]，以及在第 1.7 节中列出的其他参考书籍。

本书反映了作者在决策/控制与运筹学领域的学术背景，这一背景也贯穿于全书的术语选择、符号体系及数学表达风格中。另一方面，强化学习方法的发展横跨人工智能领域、决策/控制领域以及运筹学领域。尽管这些领域所研究问题的数学结构存在相似性，但术语体系、符号习惯与研究文化存在显著差异，可能令刚进入该领域的研究人员感到困惑。为此，我们在第 1.6.1 节特别设置了术语对照表与领域导读，以帮助读者全面理解动态规划/强化学习领域的文献体系。

1.1 阿尔法零、离线训练与在线执行

强化学习领域中最令人兴奋的成功案例之一是 DeepMind 公司开发的阿尔法围棋（AlphaGo）和阿尔法零（AlphaZero）程序；参见 [SHM+16]、[SHS+17]、[SSS+17]。阿尔法零能够玩国际象棋、围棋及其他游戏，并且在性能和通用性方面优于阿尔法围棋，而阿尔法围棋仅限于围棋。到 2022 年，这两个程序的水平都超过了所有竞争对手的计算机程序，并且远远超越了所有人类棋手。这些程序在其他方面也极为出色。特别是，它们无需人类指导，仅通过与自己对弈生成的数据就学会了如何下棋。此外，它们的学习速度极快。例如，AlphaZero 仅用了数小时（当然，这得益于强大的并行计算能力）就学会了下国际象棋，并且达到了超越所有人类和计算机程序的水平。

也许最令人印象深刻的方面在于，阿尔法零下棋不仅更强，其长远战略眼光也与人类截然不同。令人称奇的是，阿尔法零发掘出了人类经过数百年深入研究后未曾发现的全新下法。尽管阿尔法零成就卓著、实现精妙，但它依然建立在久经考验的理论和方法论基础上，而这正是本书的主题，同时这些方法也可广泛应用于工程、经济学及其他领域。这里所涉及的方法包括动态规划、策略迭代、有限前瞻、策略前展以及在值空间的近似方法。

值得注意的是，阿尔法零设计原理与 Tesauro 在计算机双陆棋方面的工作 [Tes94]、[Tes95]、[TG96] 有许多共同之处。Tesauro 的程序在 1990 年代中期激发了人们对强化学习的极大兴趣，该程序同样展现出与人类双陆棋选手截然不同且更为出色的下法。另一个令人印象深刻的程序是用于（单人）俄罗斯方块游戏的程序，该程序同样基于策略迭代方法，由 Scherrer 等人在 [SGG+15] 中提出，其工作借鉴了多项早期研究成果，包括 Tsitsiklis 和 Van Roy [TVR96]、以及 Bertsekas 和 Ioffe [BI96] 的研究。为了更好地理解阿尔法零和阿尔法围棋-零（AlphaGo Zero）[1]与 Tesauro 的程序及此处所述概念之间的联系，推荐阅读论文 [SSS+17] 中的“方法”部分。

为了理解阿尔法零的整体结构及其与我们的动态规划/强化学习方法论的联系，我们可以将其设计分为两部分：离线训练（*off-line training*）和在线执行（*on-line play*）。离线训练部分是一种算法，用于学习如何评估棋局以及如何利用一个默认/基本的棋手引导自己走向良好的局面；而在线执行部分则是另一种算法，它利用离线训练中获得的知识，在与人类或计算机对手实时对弈时生成优秀的走法。接下来我们将简要描述这些算法，并将它们与动态规划的概念和原理联系起来。

[1]阿尔法围棋-零是 DeepMind 公司在阿尔法围棋和阿尔法零之间发布的用于围棋的程序。——译者注

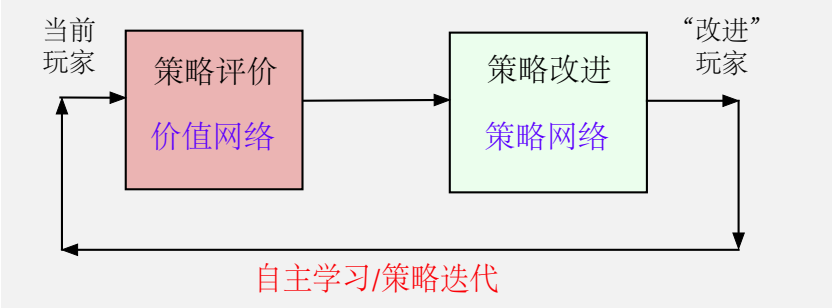


图 1.1.1: 阿尔法零训练算法示意图。该算法生成一系列棋局评价器和棋手。棋局评价器和棋手分别由两个神经网络表示：价值网络和策略网络，它们都以棋局作为输入，并分别输出棋局评价值和下一步走法概率。

离线训练与策略迭代

这是程序中通过离线自我训练学习如何下棋的部分，如图 1.1.1所示。该算法生成一系列的棋手和棋局评价器。棋手为给定局面中所有合法走法赋予“概率”，代表每个走法被选择的可能性；而棋局评价器则为给定局面赋予一个数值分数，用以预测在该局面下玩家获胜的机会。棋手和棋局评价器分别由两个神经网络表示，即策略网络和价值网络，它们都以棋局作为输入，并分别生成一组走法概率和一个棋局评价值。[2]

如果采用更传统的动态规划术语，棋局代表了游戏的状态；棋局评价器则是一个费用函数，它给出（或估计出）在给定状态下的最优费用；而棋手则是在给定状态下选择动作/控制的随机策略。[3]

整体而言，该训练算法可以看作某种形式的策略迭代（*policy iteration*），即一种经典的动态规划算法，也是本书中我们主要关注的内容。从一个初始棋手出发，该算法不断生成（近似）改进的棋手，并最终确定一个经实验证明在所有生成的棋手中“最好”的棋手。[4]策略迭代在概念上可以分为两个阶段（参见图 1.1.1）。

(a) 策略评价（*policy evaluation*）：给定当前棋手和一个棋局，从该棋局作为出发点的一局对弈的结果就提供了一个数据点。收集大量数据点后，这些数据被用于训练一个价值网络，而该网络随后充当棋手的局面评价器。

(b) 策略改进 *(policy improvement)*：给定当前棋手及其棋局评价器，从许多棋局开始，尝试选择和评估试探性走法序列，直到游戏结束。然后，通过将当前棋手的走法概率向那些获得最佳结果的试探性走法进行调整，从而生成一个改进的棋手。在阿尔法零中，

[2]这里神经网络扮演着函数近似器（function approximator）的角色；详见第 3 章。如果我们将棋手看作一个为棋局分配走法概率的函数，并将棋局评价器看作一个为棋局分配数值评分的函数，那么策略网络和价值网络便是通过数据训练提供了对这些函数的近似（关于神经网络及其他逼近架构的训练算法，也可参见强化学习书籍 [Ber19b]、[Ber20b] 以及神经元动态规划书籍 [BT96] 的讨论）。

[3]另一个复杂之处在于，国际象棋和围棋都是双人博弈，而本书的大部分内容都只涉及单人优化问题。然而，动态规划理论同样可以扩展到双人博弈，尽管我们不会将重点放在这一扩展上。或者，我们也可以考虑训练一个博弈程序，使其能够与一个已知的固定对手对战；这是一种单人环境的设置，在第 4.7.3 节中会有进一步讨论。

[4]引用论文 [SSS+17] 的原文：“阿尔法围棋-零自主对弈算法同样可以理解为一种近似的策略迭代方法，其中的蒙特卡罗树搜索既用于策略改进，也用于策略评价。策略改进从一个神经网络策略开始，基于该策略的建议执行蒙特卡罗树搜索，然后将（更强大的）搜索策略投影回神经网络的函数空间。策略评价则应用于（更强大的）搜索策略：自主对弈游戏的结果也同样被投影回神经网络的函数空间。这些投影步骤通过训练神经网络参数，使所得神经网络分别匹配搜索概率和自我对弈游戏的结果来实现。”但需要注意的是，一个通过自我对弈训练得到的双人博弈玩家可能会在面对能够利用训练漏洞的特定人类或计算机玩家时遭遇失败。这在理论上是可能的，但很罕见；详见我们在第 4.7 节中的讨论。

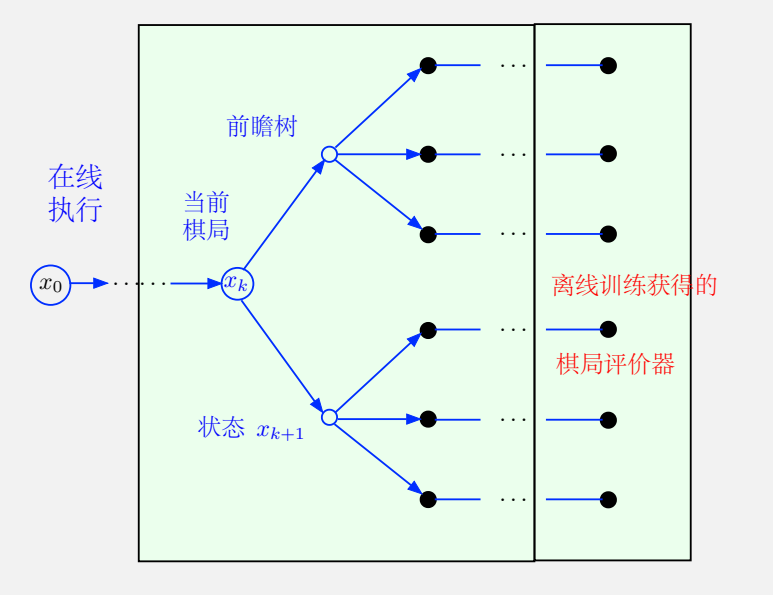


图 1.1.2: 阿尔法零训练以及许多其他计算机国际象棋程序中在线棋手的示意图。在给定棋局下，该算法生成一棵多步走法前瞻树，直至某个预定深度，并利用离线棋手的棋局评价器对剩余走法的效果进行评估。这里涉及许多我们未讨论的实现细节；例如，前瞻是选择性的，因为部分前瞻路径会通过某种形式的蒙特卡罗树搜索被剪枝。请注意，阿尔法零离线训练得到的神经网络同时产生棋局评价器和棋手策略，但神经网络训练得到的策略并不直接用于在线执行。

这一步是通过一种称为*蒙特卡罗树搜索*（*Monte Carlo Tree Search*）的复杂算法完成的。然而，策略改进也可以通过更简单的方法实现。例如，可以尝试从给定局面出发的所有可能的走法序列，向前延伸几步，并由当前棋手的棋局评价器对终局进行评估。通过这种方式获得的走法评估结果可用来调整当前棋手的走法概率，使其趋向于更成功的走法，从而获得用于训练代表新棋手的策略网络的数据。

Tesauro 的 TD-Gammon 算法 [Tes94] 同样基于近似策略迭代，但采用了一种不同的近似策略评价方法，该方法基于 TD(λ) 算法。与阿尔法零不同，TD-Gammon 并不使用策略网络或蒙特卡罗树搜索。相反，它完全依赖于价值网络，通过在线执行一步或两步前瞻的最小化来生成走法，从而复制策略网络的功能。详细描述请参见神经元动态规划 [BT96] 的第 8.6 节。

在线执行与值空间近似——策略前展

假设通过上述阿尔法零的离线训练过程获得了一个“最终”棋手。原则上，该棋手可以利用离线训练得到的策略网络生成走法概率，从而与任何人类或计算机对手进行国际象棋比赛。在任何局面下，该棋手只需选择策略网络中概率最高的走法即可。虽然这种方法可以使棋手迅速做出决策，但其实力不足以击败技术精湛的人类棋手。阿尔法零的非凡实力正是在将离线训练得到的棋手和棋局评价器嵌入到另一算法中后才得以展现，我们所得的棋手为为在线棋手（见图 1.1.2）。在某个棋局下，在线棋手会生成一个包含所有可能的多步走法和应对走法序列的前瞻树，直至达到预设深度。随后，它利用离线获得的价值网络中的棋局评价器对剩余走法的效果进行评价。

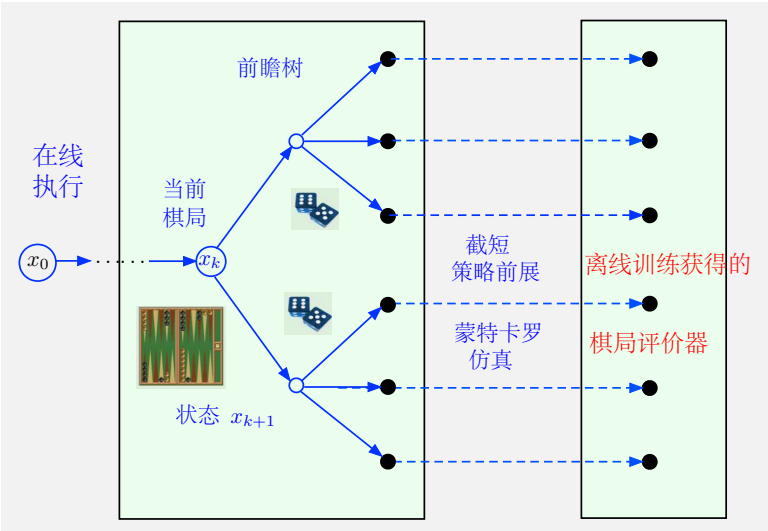


图 1.1.3: 这是带有截短策略前展的 TD-Gammon 架构示意图 [TG96]。该程序采用相对较短的前瞻最小化，随后进行策略前展和终局棋局评价（即使用一步前瞻棋手进行游戏仿真；经过若干步后，游戏被截断，并对截断时对棋局执行评价）。请注意，双陆棋涉及随机不确定性，其状态由当前棋盘局面和骰子点数组成。因此，策略前展需要蒙特卡罗模拟，这非常耗时。结果是在严格时间限制下进行对弈的西洋双陆棋程序采用了有限或高度截短形式的策略前展。

Tesauro 的 TD-Gammon 程序最终版本 [TG96] 的架构与阿尔法零类似，都采用了离线神经网络训练得到的终局局面评价器；见图 1.1.3。然而，它还包含一个中间部分，称为“截短策略前展（truncated rollout）”，即在使用棋局评价器之前，先根据给定棋手执行几步仿真。实际上，策略前展可以看作是一种经济地延长前瞻深度的方法。在已公开的阿尔法零/国际象棋版本 [SHS+17] 中，也包含策略前展部分，但其实现较为粗糙；因为第一部分（多步前瞻）已经相当长且高效，所以一个复杂的策略前展部分并非必需。策略前展在阿尔法围棋中 [SHM+16] 扮演着重要角色，并且在 Tesauro 的双陆棋程序中至关重要，因为在该程序中，由于前瞻树迅速扩展，很长的多步前瞻是不可行的。

与阿尔法零和 TD-Gammon 类似的架构在本书中统称为值空间近似（*approximation in value space*）。这种类型的架构也被称为近似动态规划（*approximate dynamic programming*）或神经元动态规划（*neuro-dynamic programming*），并将成为我们讨论的核心内容。[5]

在其他一些场合，值空间近似方法也被用于控制系统设计，尤其是在模型预测控制（*model predictive control*）（英文简称 MPC）中，该方法在第 2.11 节中有简要描述。在此方法中，前瞻最小化的步数被称为控制区间（*control interval*），而前瞻最小化与截短策略前展总步数被称为预测区间（*control interval*）；参见例如 Magni 等人的工作 [MDNMS01]。[6]在模型预测控制中，截短策略前展作为较长的前瞻最小化的经济替代方案的优势是广为人知的。

需要注意的是，前面对阿尔法零以及相关的在值空间近似架构的描述有些简化。随着

[5] “近似动态规划”和“神经元动态规划”这两个名称常常被用作强化学习的同义词。然而，通常认为强化学习还包括策略空间近似的方法，即在一组参数化策略中搜索最优参数。这个搜索是通过与动态规划大体无关的方法来完成的，例如随机梯度或随机搜索方法。策略空间近似可用于离线设计一个策略，该策略随后可用于在线策略前展。本书对此将进行较为简要的讨论（参见第 3.4 节和第 3.5 节）。更详细的讨论（使用与此处一致的术语）可以参见强化学习书籍 [Ber19b] 的第 5 章。

[6] Matlab 的模型预测控制设计工具包明确允许用户设置这两个区间。

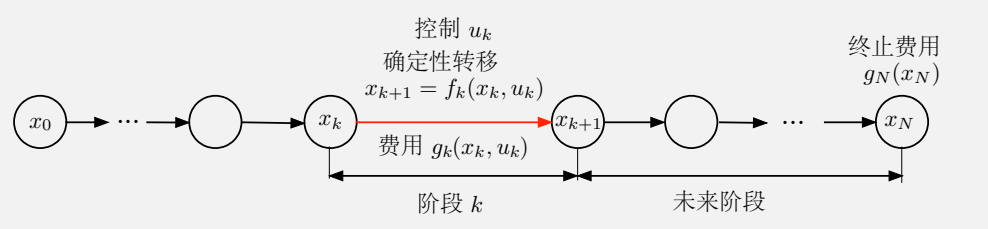


图 1.2.1: 一个确定性的 N 阶段最优控制问题。从 x_k 出发，在控制信号 u_k 的作用下，下一时刻的状态依据系统方程

$$x_{k+1} = f_k(x_k, u_k)$$

完全确定，并产生一个取值为 $g_k(x_k, u_k)$ 的阶段费用。

本书的展开，我们将讨论更多的改进和细节。然而，基于费用函数近似的动态规划思想（类似于图 1.1.2中所示的在线棋手）将是本书的核心内容。此外，离线训练与在线策略实现之间的算法划分在我们的讨论中也具有概念上的重要意义。

我们最后指出，在值空间近似中，离线训练和在线执行算法往往可以解耦并独立设计。例如，离线训练部分可能相对简单，比如使用已知策略进行策略前展，而不进行截短或使用终止费用近似。反过来，也可以采用复杂的流程对终止费用函数进行离线训练，这一近似在值空间近似方案中会紧接着一步或多步前瞻之后使用。

1.2 确定性动态规划

在所有动态规划问题中，核心对象是一个离散时间动态系统，该系统在动作或控制的作用下生成一系列状态。该系统可以以确定性或随机（在额外随机扰动的影响下）方式演化。

1.2.1 有限阶段问题

在有限阶段问题中，系统会经历 N 个时刻（也称为阶段）的演化。在 k 时刻，系统的状态和所受到的控制分别记作 x_k 和 u_k。在确定性系统中，x_{k+1} 不是随机生成的，即它由 x_k 和 u_k 完全决定；参见图1.2.1。因此，一个确定性动态规划问题涉及如下离散时间的动态系统

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1, \tag{1.1}$$

其中，

k 是时刻；

x_k 是系统状态，属于某种空间的一个元素；

u_k 是控制或决策变量，它属于某个给定的集合 $U_k(x_k)$，该集合由 x_k 决定，并可随时刻 k 而变化；

f_k 是关于 (x_k, u_k) 的函数，用于描述系统状态从 k 时刻到 $k+1$ 时刻的演化机制；

N 是时域或决策可作用于系统的时刻的总数。

在状态数量有限的情况下，系统函数 f_k 可以用一个表格来表示，该表格为每个可能的状态-控制对 (x_k, u_k) 给出下一个状态 x_{k+1}。否则，就需要用数学表达式或计算机实现来表示 f_k。

我们称所有可能的 x_k 组成的集合为 k 时刻的状态空间（*state space*）。状态空间可以是任意类型的集合，并且可以随着时刻 k 而发生改变。类似地，由所有可能的 u_k 构成的集合被称为控制空间（*control space*）。同理，U_k 可以是任意类型的集合，并且可能随着时刻 k 的不同而改变。[7]

除上述概念外，动态规划问题还会涉及费用函数这一概念。我们记 k 时刻所产生的阶段费用为 $g_k(x_k, u_k)$，那么通过把阶段费用从当前时刻起累加起来就定义了当前状态的费用函数。这里的 $g_k(x_k, u_k)$ 是一个关于 (x_k, u_k) 的函数，其输出为实数，且函数本身可能随时刻 k 而发生变化。对于一个给定的初始状态 x_0，控制序列 $\{u_0, \ldots, u_{N-1}\}$ 的总费用为

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{1.2}$$

其中 $g_N(x_N)$ 为系统演化终点的终止费用。上述总费用的定义是完备的，这是因为控制序列 $\{u_0, \ldots, u_{N-1}\}$ 和初始状态 x_0 通过式(1.1)共同确定了状态序列 $\{x_1, \ldots, x_N\}$。我们希望在所有满足控制约束的决策序列 $\{u_0, \ldots, u_{N-1}\}$ 中寻找能最小化费用(1.2)的序列，从而得到最优值[8]

$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0,\ldots,N-1}} J(x_0; u_0, \ldots, u_{N-1}) \tag{1.3}$$

所定义的关于 x_0 的函数。

离散最优控制问题

许多情况下，状态与控制空间本身就是由有限多的离散元素构成的。在此类问题中，从任意 x_k 到可能的 x_{k+1} 的系统状态转移都可以很方便地用有向无环图来描述。图中的节点代表系统的状态，而边则代表状态动作对 (x_k, u_k)。以 x_k 为起点的每个边对应于控制约束集 $U_k(x_k)$ 中的一个控制选项，并且其终点为下一时刻状态 $f_k(x_k, u_k)$，见图1.2.2。为了有效采用图来处理该问题的最终阶段，我们在图中添加一个虚拟终止节点 t。每个 N 阶段的状态 x_N 与终止节点 t 之间被一个费用为 $g_N(x_N)$ 的边相连。

注意到每个控制序列 $\{u_0, \ldots, u_{N-1}\}$ 都对应于以初始状态（阶段 0 的状态 s）为起点并终止于 N 阶段某状态的一条路径。如果我们把一条边对应的费用视作其长度，那么我们容易发现求解一个确定性的有限状态有限阶段问题就等价于寻找其相应转移图中以初始节点 s 为起点、以终止节点 t 为终点的最小长度（最短）的路径（*a deterministic finite-state*

[7]这种通用性是动态规划方法的一大优势，也决定了本书以及作者其他动态规划著作的论述风格。通过允许通用的状态和控制空间（可以是离散的、连续的或两者的混合），以及允许这些空间随 k 发生变化，我们能够把注意力集中在动态规划方法真正关键的算法方面，排除模型中那些无关紧要的假设和限制，从而避免重复分析。

我们动态规划模型的通用性也是我们选择这种符号系统的部分原因。在人工智能和运筹学领域，有限状态模型，即所谓马尔可夫决策问题（Markovian Decision Problem，英文简写作 MDP），非常常见，并采用转移概率符号（参见第 1.7.2 节）。不幸的是，这种符号表示法不太适用于确定性模型以及连续空间模型，而这两类模型对于本书的讨论都非常重要。对于后者，它涉及连续空间上的转移概率分布，从而导致相应的数学表述比使用系统函数(1.1)的方法要复杂得多，也不直观。

[8]本书 (在应使用 “inf” 的位置) 全部使用 “min” 来表示在一个可行决策集上取得的最小值。即使当我们不确定上述最小值能否通过某一可行的决策取得时，我们仍会采用符号 “min”。

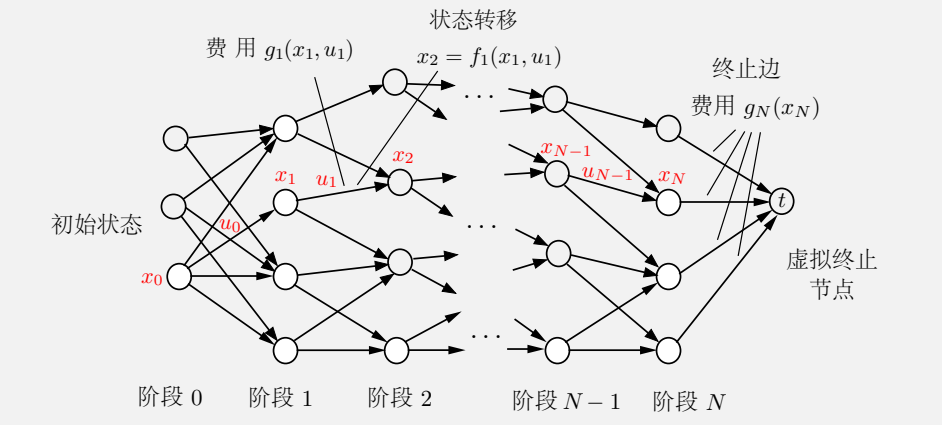


图 1.2.2: 一个确定性有限状态问题的转移图。图中的节点代表状态 x_k，边则代表状态决策对 (x_k,u_k)。边 (x_k,u_k) 以 x_k 和 $f_k(x_k,u_k)$ 作为起点和终点。我们把相应状态转移的费用 $g_k(x_k,u_k)$ 视作该边的边长。这个确定性有限状态的问题就等价于寻找从初始节点 s 到终止节点 t 的最短路径。

finite horizon problem is equivalent to finding a minimum-length (or shortest) path from the initial node s of the graph to the terminal node t)。此处路径的长度指的是构成路径的边的长度之和。[9]

一般而言，组合优化问题可以表述为确定性的、有限状态、有限阶段的最优控制问题。其核心思想是将解分解为若干部分，并依次进行计算。我们用以下调度问题作为示例。

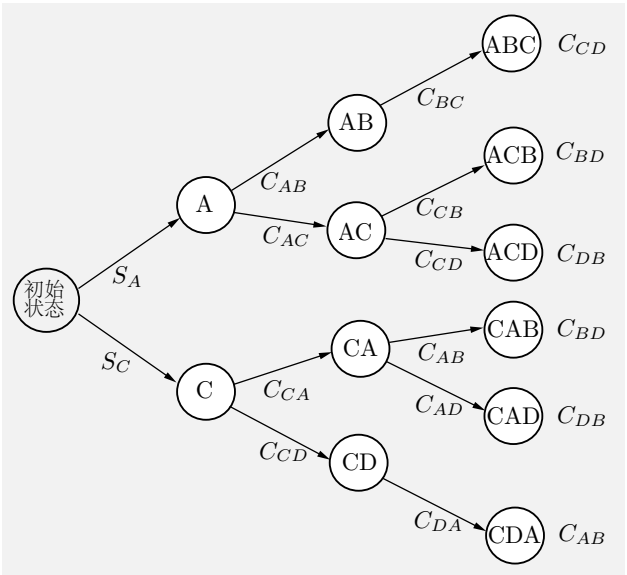


图 1.2.3: 例1.2.1所述的确定性调度问题的转移图。图中的每条边都代表了把某个状态（边的起点）导向另一状态（边的终点）的一个决定。边所对应的费用标注在其一旁。最终工序所对应的费用则作为终止费用标注在图中的终止节点的一旁。

例 1.2.1 (一个确定性调度问题). 假设为了生产某产品，在某机器上需要执行四道工序。这

[9]事实上，（在包括有环的任何有向图中）求解最短路径问题都可以转化为有限状态的确定性最优控制问题。更多与本节内容相关的、关于最短路径求解方法的详细介绍，参见 [Ber17] 的 2.1 节，[Ber91] 和 [Ber98]。

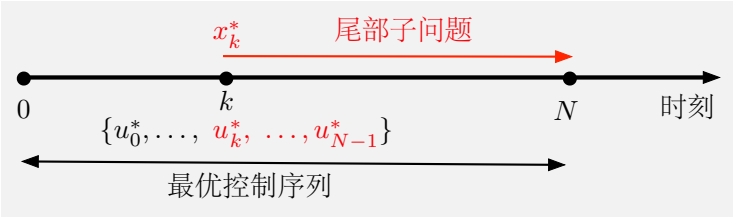


图 1.2.4: 最优性原理图示。对于一个最优控制序列 $\{u_0^*, \ldots, u_{N-1}^*\}$，其尾部 $\{u_k^*, \ldots, u_{N-1}^*\}$ 是一个新的最优控制问题的最优控制序列。我们记原序列相应的最优轨迹为 $\{x_1^*, \ldots, x_N^*\}$，那么这个新的问题就是以该最优轨迹中的状态 x_k^* 为初始状态的原问题的尾部子问题。

四道工序以字母 A、B、C 和 D 来表示。我们假设，工序 B 需在工序 A 之后执行，工序 C 需在工序 D 之前执行。(因此，工序序列 CDAB 是允许的，而 CDBA 则不允许。) 从工序 m 到 n 的编排费用 C_{mn} 给定。此外，从工序 A 和 C 开始分别需要的启动费用是 S_A 和 S_C (见图*1.2.3*)。一个工序序列的费用是该序列对应的阶段费用之和；例如，工序序列 ACDB 的费用为

$$S_A + C_{AC} + C_{CD} + C_{DB}.$$

我们可以把该问题视作关于三个决策序列的问题，即安排执行前三道工序 (第四道工序由前三道确定)。因此，将已经安排的工序序列作为状态就是一个合理的选择，相应的，初始状态就是一个人为引入的状态，用来表示决策过程的开端。图*1.2.3*给出了该问题对应的状态及决策的所有可能的状态转移。此处的问题是确定性的，即在任意一个给定的状态下任选一个控制，系统下一时刻的状态都是唯一确定的。例如，在状态 AC 时决定安排工艺 D，那么下一时刻状态即为 ACD，且该决策花费 C_{CD} 的费用。因此，上述问题可以很方便地用图*1.2.3* 所示的转移图来说明。图中的每一条以初始状态为起点、以某一终止时刻的状态为终点的路径即为原问题的一个解。这一解的费用就是构成该路径的所有边的费用与终止费用之和。该问题的最优解就对应于相应转移图中费用最小的上述路径。

1.2.2 动态规划算法

在本节中，我们将介绍动态规划算法并证明其合理性。一般来说，动态规划用于解决跨越 N 个阶段的序贯决策问题，通过将其分解为一系列更简单的单阶段问题来求解。

具体来说，该算法旨在通过生成一系列的最优费用函数 $J_0^*, \ldots, J_N^*$ 来求解一系列最优控制 $u_0^*, \ldots, u_{N-1}^*$。它以 J_N^* 等于终止费用函数 g_N 开始，然后通过求解一个优化变量为 u_{N-1} 的单阶段决策问题来计算下一个函数 J_{N-1}^*。接着，它利用 J_{N-1}^* 来计算 J_{N-2}^*，并以类似的方式依次计算所有剩余的费用函数 $J_{N-3}^*, \ldots, J_0^*$.

这种算法基于一个简单的概念，即最优性原理 (*principle of optimality*)。现将该原理概述如下；见图1.2.4。

最优性原理. 给定初始状态 x_0，记相应的某一最优控制序列为 $\{u_0^*, \ldots, u_{N-1}^*\}$。在该控制序列作用于状态方程(1.1)后，系统的状态序列为 $\{x_1^*, \ldots, x_N^*\}$。现考虑如下子问题：从 k 时刻的状态 x_k^* 出发，以 $\{u_k, \ldots, u_{N-1}\}$，$u_m \in U_m(x_m)$，$m = k, \ldots, N-1$ 为优化变量，我

们希望最小化从 k 时刻到 N 时刻累积的“展望费用”

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N).$$

那么，该子问题的最优解是 $\{u_k^*, \ldots, u_{N-1}^*\}$，即通过截短原问题的最优控制所得的序列。

上述的子问题被称为始于 x_k^* 的尾部子问题（*tail subproblem*）。简单来讲，最优性原理就是指出一个最优控制序列的尾部即为其相应的尾部子问题的最优解（*the tail of an optimal sequence is optimal for the tail subproblem*）。从直观上讲，最优性原理的论证很容易。假设截取的控制序列 $\{u_k^*, \ldots, u_{N-1}^*\}$ 不是相应子问题的最优解。那么在求解原问题的过程中，当到达状态 x_k^* 时，通过将剩余的控制序列替换为以上假设的子问题的最优解（因为到达状态 x_k^* 前采用的控制序列 $u_0^*, \ldots, u_{k-1}^*$ 并不会约束后续的控制选择，所以这么做是可行的），我们将得到比原最优解更优的控制序列。

如果以驾车出行来类比，假设从凤凰城到波士顿的最快路径途经圣路易斯。那么最优性原理就是如下显然的事实：以上最快路径中从圣路易斯到波士顿的部分也是从圣路易斯开始到波士顿结束的最快路径。[10]

最优性原理表明我们可以从后往前地、逐步地求解最优费用函数：首先针对仅涉及最后一段的“尾部子问题”求解其最优费用函数，然后求解涉及最后两个阶段的“尾部子问题”，依此类推，直到求得整个问题的最优费用函数。

动态规划算法就是基于上述思想：该算法依序执行，通过利用已知的更短时间跨度的尾部子问题的解，解决某一给定的时间跨度的所有尾部子问题（*solving all the tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length*）。我们以例1.2.1中介绍的调度问题来说明上述算法。以下例子中的计算比较繁杂，且跳过这些计算步骤并不影响对后续内容的理解。然而，对于动态规划算法的初学者，通读这些计算步骤仍然是有益的。

例 1.2.1 （一个确定性调度问题——续）**.** 现在我们考虑例*1.2.1*中介绍的调度问题，并利用最优性原理求解。我们需要以最优的方式来安排工序 A、B、C 和 D。在两个工序交接时会产生一定的转移费用，其相应数值标注在图*1.2.5*中对应边上。

根据最优性原理可知，原问题的一个最优调度序列的“尾部”必然是相应的尾部子问题的一个最优解。例如，假设最优调度为 CABD。那么在将前两道工序 C 和 A 确定后剩余两道工序的安排应当是先 B 后 D，即 BD，而不是 DB。考虑到这一点，我们先求解所有长度为二的尾部子问题，接着再求解长度为三的尾部子问题，最后求解长度为四的原问题（对于长度为一的子问题，因为已经安排了三道工序，最后一个决策即为仅有的剩余工序，显然无需求解）。我们后续会看到，一旦长度为 k 的子问题得到解决，那么长度为 $k+1$ 的问题将很容易求解，这就是动态规划算法的核心。

长度为 2 的尾部子问题：这些子问题涉及到两个没有安排的工序，并且对应于状态 AB、AC、CA 和 CD（见图*1.2.5*）。

状态 AB：从该状态出发，接下来唯一可安排的工序为 C，因此该子问题的最优费用为 9（在工序 B 后安排 C 花费 3，C 后安排 D 花费 6，两费用相加可得）。

[10] 正如贝尔曼 [Bel57] 所言：“一条最优轨迹具有这样的特性：在某个中间点上，无论是如何到达该点的，其余的轨迹部分都必须与从该中间点出发所计算的最优轨迹相一致。”

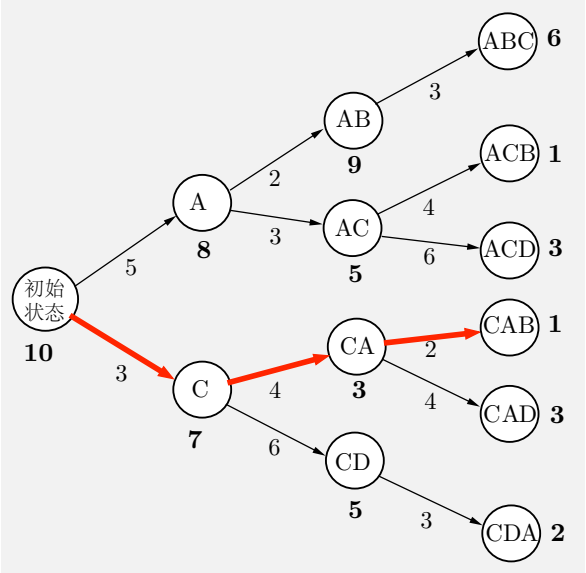


图 1.2.5: 确定性调度问题图示。每个决策的费用标注在相应的边上。每个节点/状态一旁标注的数值代表从该状态为起点进行工序安排的最优解所需的费用。该费用即为相应尾部子问题的最优费用（参照最优性原理）。原问题的最优费用为 10，并标注在初始状态一旁。最优调度由加粗的边标出。

状态 AC：此处有两种调度方案（a）先安排 B 再安排 D，共计花费 5，或者（b）先安排 D 再安排 B，共计花费 9。两选项中前者更优，因此该尾部子问题的最优费用为 5，并注于图*1.2.5*中节点 AC 旁边。

状态 CA：此处有两种调度方案（a）先安排 B 再安排 D，共计花费 3，或者（b）先安排 D 再安排 B，共计花费 7。两选项中前者更优，因此该尾部子问题的最优费用为 3，并注于图*1.2.5*中节点 CA 旁边。

状态 CD：从该状态出发，接下来唯一可安排的工序为 A，因此该问题的最优费用为 5。

长度为 3 的尾部子问题：现在我们可以通过求得的长度为 *2* 的子问题的最优解来求解这些子问题。

状态 A：从该状态出发可选的调度安排有（a）B（花费 2）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（如前所述，其最优费用为 9），因而总计花费 11，或者（b）先安排 C（花费 3）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 5，如前所述），因而总计花费 8。上述的第二种调度方案是最优的，其相应的费用 8 即为该尾部子问题的最优费用，因而标注于图*1.2.5*中节点 A 旁边。

状态 C：从该状态出发可选的调度安排有（a）A（花费 4）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 3，见前述计算），因而总计花费 7，或者（b）先安排 D（花费 6）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 5，见前述计算），因而总计花费 11。上述的第一种调度方案

是最优的，其相应的费用 7 即为该尾部子问题的最优费用，因而标注于图*1.2.5*中节点 C 旁边。

长度为 4 的原问题：从初始状态出发可选的调度安排有（a）A（花费 5）并从所得状态为起点求解相应的长度为 3 的子问题的最优解（如前所述，其最优费用为 8），因而总计花费 13，或者（b）先安排 C（花费 3）并从所得状态为起点求解相应的长度为 3 的子问题的最优解（如前所述，其最优费用为 7），因而总计花费 10。上述的第二种调度方案是最优的，其相应的费用 10 即为原问题的最优费用，因而标注于图*1.2.5*中初始状态节点旁边。

在利用所有尾部子问题的解计算出原问题的最优费用后，我们可以求得最优调度：我们从初始状态出发，从前向后计算，每一时刻选择最优的调度，即选择从当前状态出发的相应尾部子问题的最优调度的第一步安排。通过这种方式，从图*1.2.5*标注可知，CABD 为最优调度。

通过动态规划求解最优控制序列

现在，我们将通过把以上的论证转化成严谨的数学表述，从而给出求解确定性有限阶段问题的动态规划算法。该算法按照顺序，从 J_N^* 出发，从后往前求解 J_{N-1}^*，J_{N-2}^* 等，从而依次构造如下函数

$$J_N^*(x_N),\, J_{N-1}^*(x_{N-1}),\ldots,J_0^*(x_0).$$

我们将说明，$J_k^*(x_k)$ 表示从 k 阶段的状态 x_k 开始的尾部子问题的最优费用。

针对确定性有限阶段问题的动态规划算法. 首先令以下方程成立

$$J_N^*(x_N)=g_N(x_N),\quad \text{对所有}x_N, \tag{1.4}$$

对 $k=0,\ldots,N-1$，令

$$J_k^*(x_k)=\min_{u_k\in U_k(x_k)}\Big[g_k(x_k,u_k)+J_{k+1}^*\big(f_k(x_k,u_k)\big)\Big],\quad \text{对所有}x_k. \tag{1.5}$$

图 1.2.6 展示了动态规划算法以及函数 $J_k^*(x_k)$ 的构建过程。注意到在阶段 k，动态规划算法要求对所有的 x_k 通过式(1.5)计算 $J_k^*(x_k)$ 后，才能进行下一步 $k-1$ 阶段的计算。动态规划算法的关键是，对每个初始状态 x_0，上述算法最后一步得到的数值 $J_0^*(x_0)$ 等于最优费用 $J^*(x_0)$。而且，我们可以证明有如下更一般的结论成立，即对于所有的 $k=0,1,\ldots,N-1$，以及 k 时刻的所有状态 x_k，如下等式成立

$$J_k^*(x_k)=\min_{\substack{u_m\in U_m(x_m)\\ m=k,\ldots,N-1}} J(x_k;u_k,\ldots,u_{N-1}), \tag{1.6}$$

其中

$$J(x_k;u_k,\ldots,u_{N-1})=g_N(x_N)+\sum_{m=k}^{N-1}g_m(x_m,u_m), \tag{1.7}$$

即 $J_k^*(x_k)$ 是始于时刻 k 以 x_k 为初始状态而终于时刻 N 的 $(N-k)$ 阶段尾部子问题的最优费用。[11]基于这一事实，我们称 $J_k^*(x_k)$ 为在状态 x_k 与时刻 k 的最优展望费用（*optimal*

[11]我们可以通过归纳法证明上述结论。已知

$$J_N^*(x_N)=g_N(x_N),$$

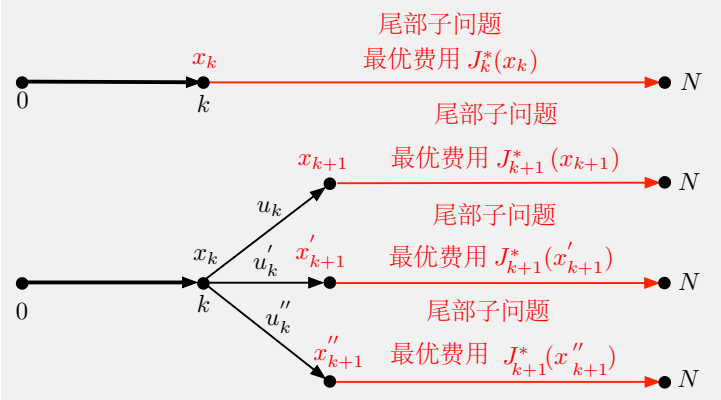


图 1.2.6: 动态规划算法示意图。对于从时刻 k 状态 x_k 开始的尾部子问题，其目标是在 $\{u_k, \ldots, u_{N-1}\}$ 上最小化从阶段 k 到 N 的“展望费用”，即

$$g_k(x_k, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N).$$

为了解决该问题，我们选择 u_k 从而使得“第一阶段费用费用与尾部子问题费用之和”最小化，即

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big].$$

cost-to-go)，并称 J_k^* 为时刻 k 的最优展望费用函数（*optimal cost-to-go function*）或者最优费用函数（*optimal cost function*）。在求解最大值的问题中，上述动态规划算法式(1.5)中的最小化运算被最大化运算所取代，而 J_k^* 也被称为 k 时刻的最优价值函数。

一旦取得函数 $J_0^*, \ldots, J_N^*$ 后，对给定的初始状态 x_0，我们可以采用以下前向的算法构造最优控制序列 $\{u_0^*, \ldots, u_{N-1}^*\}$ 及其相应的状态轨迹 $\{x_1^*, \ldots, x_N^*\}$。

构造最优控制序列 $\{u_0^*, \ldots, u_{N-1}^*\}$**.** 首先令

$$u_0^* \in \arg\min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + J_1^*\big(f_0(x_0, u_0)\big) \Big],$$

及

$$x_1^* = f_0(x_0, u_0^*).$$

依次向后，对 $k = 1, 2, \ldots, N-1$，令

$$u_k^* \in \arg\min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \tag{1.8}$$

故上述结论在 $k = N$ 时成立。为证明对所有的 k 都成立，我们据式(1.6)和(1.7)得到如下关系

$$\begin{aligned} J_k^*(x_k) &= \min_{\substack{u_m \in U_m(x_m) \\ m=k,\ldots,N-1}} \Big[g_N(x_N) + \sum_{m=k}^{N-1} g_m(x_m, u_m) \Big] \\ &= \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + \min_{\substack{u_m \in U_m(x_m) \\ m=k+1,\ldots,N-1}} \Big[g_N(x_N) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) \Big] \Big] \\ &= \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \end{aligned}$$

其中最后一个等式是基于归纳假设而成立。此处有一个数学上很微妙的细节。当求最小值的时候，在某些状态 x_k，展望费用函数 J_k^* 可能取值 $-\infty$。即便如此，上述归纳法中的论述依然成立。

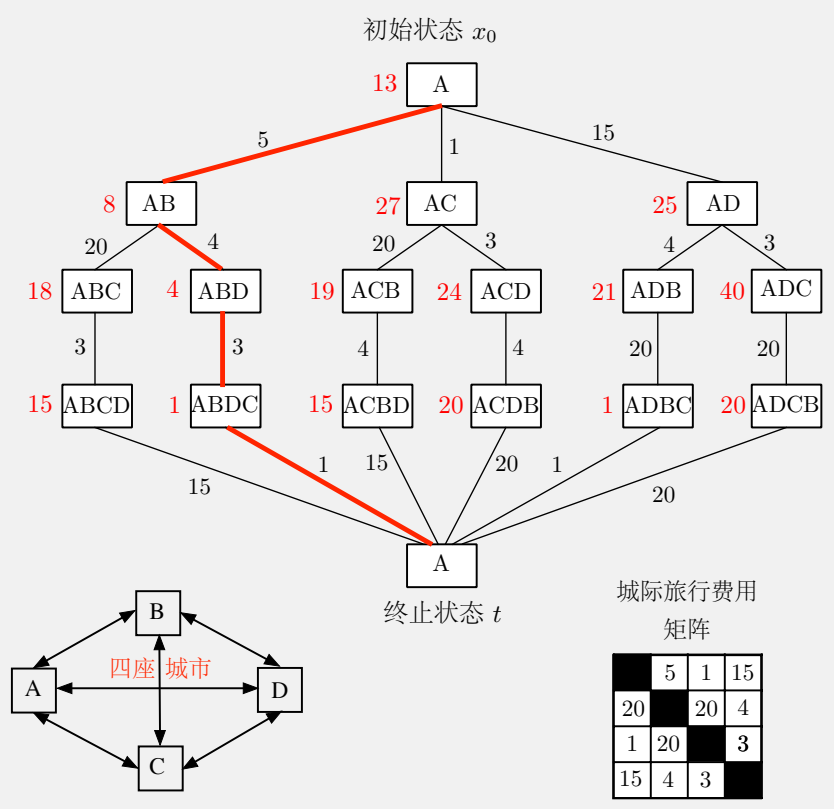


图 1.2.7: 将旅行商问题转化为一个动态规划问题的示例。四座城市 A、B、C 和 D 之间的旅行费用标注在图底部的矩阵中。通过将每个包含 k 座城市的序列作为一个节点，我们引入一个图。每个含 k-城市的节点即为 k 阶段的一个状态。转移费用/旅行时间则标注在相应的边的一侧。最优展望费用可以通过动态规划算法给出。该算法从终止状态开始，从后往前，直到抵达初始状态为止，所得数值标注于相应节点旁。在该问题中有两个最优序列（即 ABDCA 和 ACDBA），我们在图中用粗线将其标出。这两个最优序列都可以通过起始于初始状态 x_0 的前向最小化运算获得 [参见式(1.8)]。

及

$$x_{k+1}^* = f_k(x_k^*, u_k^*).$$

同样的算法也可以用于构造任何尾部子问题的最优控制序列。图1.2.5描绘出了动态规划算法用于规划问题例1.2.1的相关计算。节点旁边的数字给出了相应的展望费用的取值，而加粗的边则标出了根据上述算法构造的最优控制序列。

下面的例子讨论了涉及 N 个城市的经典旅行商问题。在这个问题中，状态数随 N 呈指数级增长，精确动态规划所需的计算量也随之呈指数级增长。我们将在后面证明，利用策略前展算法，我们可以用计算量呈多项式增长的方式近似求解该问题。

例 1.2.2 (旅行商问题). 给定 N 座城市以及城际旅行所需的时间，在对每座城市要求有且只有一次访问后，我们需要返回出发城市。在所有可行的方案中，我们希望找出总旅行时间最小的一个规划路线。令 $k = 1, \ldots, N$，以 k 座不同城市构成的序列作为节点，我们就可以得到一个图，从而把该问题转化成一个动态规划问题。每个包含 k 座城市的序列都是 k 阶段的一个状态。初始状态 x_0 则是作为出发点的某城市（在图*1.2.7*中为城市 A）。从一个 k-城市的节点/状态出发，通过在相应序列后添加一个新的城市，系统在下一时刻抵达相应

的 $k+1$-城市的阶段/状态，并花费 $k+1$ 城市序列的最后两座城之间的城际旅行费用；见图*1.2.7*。此外，每个代表 N 座城市序列的节点都与一个虚拟终止节点 t 相连，并且相应的连接边对应的旅行费用为从序列的最后一座城市到出发城市的旅行费用。这样，我们就将旅行商问题转化成了一个动态规划问题。

从每个节点出发到终止状态的最优展望费用可以通过动态规划算法获得，其相应数值被标注在节点的一旁。然而，应当注意到，节点数量随着城市数量 N 指数递增。这就导致当 N 取大值时，动态规划算法难以执行。因此，大规模的旅行商和相关的调度问题通常使用近似方法求解，而其中的一些方法正是基于动态规划而设计的，我们也会在后续章节中对它们进一步讨论。

Q-因子和 Q-学习

动态规划算法(1.5)的另一种（且等价的）形式间接地使用了最优展望费用函数 J_k^*。具体来说，该形式针对所有的状态-控制对 (x_k, u_k) 和时刻 k 生成如下的最优 Q-因子（*optimal Q-factors*）

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big). \tag{1.9}$$

因此最优 Q-因子即是动态规划等式(1.5)右侧被最小化的表达式。[12]

我们注意到该等式表明最优费用函数 J_k^* 可以由最优 Q-因子 Q_k^* 通过以下计算获得

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k). \tag{1.10}$$

此外，通过上述等式，动态规划算法(1.5)改写作一种仅涉及 Q-因子的等价形式 [参考(1.9)-(1.10)]

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + \min_{u_{k+1} \in U_{k+1}\left(f_k(x_k, u_k)\right)} Q_{k+1}^*\big(f_k(x_k, u_k), u_{k+1}\big).$$

这种以及其他相关算法的精确和近似形式，包括针对随机最优控制问题的对应算法，构成了强化学习中一类重要的方法，即所谓的Q-学习（*Q-learning*）。

1.2.3 值空间近似与策略前展

需要注意的是，只有在通过动态规划为所有 k 和 x_k 计算出 $J_k^*(x_k)$ 后，式(1.8)中正向构造最优控制序列才成为可能。不幸的是，在实际应用中，由于可能的 x_k 和 k 数量可能非常庞大，这通常会耗费难以接受的时间。然而，如果将最优展望费用函数 J_k^* 替换为一些近似函数 $\tilde{J}_k$，则可以采用类似的正向算法构造控制序列。这正是强化学习中一个核心思想的基础：值空间的近似（*approximation in value space*）。[13]基于在动态规划算法(1.8)，采用 $\tilde{J}_k$ 替代 J_k^*，从而构造一个次优解 $\{\tilde{u}_0, \ldots, \tilde{u}_{N-1}\}$ 来代替最优解 $\{u_0^*, \ldots, u_{N-1}^*\}$.

[12] “Q-因子”这一术语在书籍 [BT96]、[Ber19b]、[Ber20b] 中均有使用，本书也沿用此术语。另一个常用术语是（在给定状态下的）“动作价值”。文献中“状态-动作价值”和“Q-值”这两个术语也很常见。“Q-因子”这一名称起源于一篇有影响力的博士论文 [Wat89] 中所采用的符号，该论文首次提出在强化学习中使用 Q 因子。

[13]值空间近似（在人工智能文献中有时称为“搜索”或“树搜索”）是一个简单的思想，早在现代强化学习方法出现之前，就已广泛应用于确定性问题。例如，从概念上讲，它构成了广泛使用的 A^* 算法的基础，该算法用于求解大规模最短路径问题的近似解。若想了解与我们近似动态规划框架相一致的 A^* 算法视角，读者可参阅作者的动态规划专著 [Ber17]。

值空间的近似——用 $\tilde{J}_k$ 代替 J_k^*. 首先令

$$\tilde{u}_0 \in \arg\min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + \tilde{J}_1\big(f_0(x_0, u_0)\big) \Big],$$

及

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0).$$

依次向后，对 $k = 1, 2, \ldots, N-1$，令

$$\tilde{u}_k \in \arg\min_{u_k \in U_k(\tilde{x}_k)} \Big[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\big(f_k(\tilde{x}_k, u_k)\big) \Big], \tag{1.11}$$

及

$$\tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k).$$

在值空间近似中，次优控制序列 $\{\tilde{u}_0, \ldots, \tilde{u}_{N\to 1}\}$ 的计算是通过前向计算完成的（一旦获得费用函数近似 $\tilde{J}_k$ 后，就不再需要反向计算）。这与最优控制序列 $\{u_0'', \ldots, u_{N\to 1}''\}$ 的计算方式类似，且独立于 $\tilde{J}_k$ 函数的具体计算方法。相对于精确动态规划算法，随机动态规划问题采用值空间近似的主要动机在于计算量的大幅缩减（在获得 $\tilde{J}_k$ 之后）：只需为系统在线控制过程中遇到的 N 个状态 $x_0, \tilde{x}_1, \ldots, \tilde{x}_{N\to 1}$ 执行最小化运算(1.11)，而不需要像精确动态规划那样在可能极其庞大的整个状态空间内对所有状态进行计算。

我们称算法(1.11)包含一步前瞻最小化（*one-step lookahead minimization*），因为它针对每个阶段 k 都求解了一个单阶段的动态规划问题。在下文中，我们还将探讨多步前瞻（*multistep lookahead*）的可能性——这种方法需要求解一个 ℓ 步动态规划问题（其中 ℓ 为整数，满足 $1 < \ell < N - k$），并以近似终止费用函数 $\tilde{J}_{k+\ell}$ 作为终止费用。在强化学习的近似方案中，多步前瞻通常（尽管并非绝对）能提供优于一步前瞻的性能表现。例如在阿尔法零国际象棋程序中，深度的多步前瞻对于实现良好的在线性能至关重要。其直观原理在于：ℓ 个阶段的费用是（通过优化的方法）“精确”处理的，那么随着 ℓ 值的增加，近似误差

$$\tilde{J}_{k+\ell} - J_{k+\ell}^*$$

的影响效应通常会逐渐减弱。然而，相较于式(1.11)对应的一步前瞻优化问题，多步前瞻对应的优化问题的求解过程需要消耗更多计算时间。

适用于确定性问题的基于基本启发式的策略前展

在值空间近似中，一个主要问题是如何构造合适的展望费用函数近似 $\tilde{J}_k$。这可以通过多种不同的方法来实现，这些方法构成了强化学习的重要组成部分。例如，$\tilde{J}_k$ 可以通过一种复杂的离线训练方法来构造，如第 1.1 节中讨论的那样。或者，也可以通过执行在线的策略前展（*rollout*）来获得 $\tilde{J}_k$，本书将对此进行详细讨论。在策略前展过程中，我们只计算所需的对应于某些状态 x_k 的费用函数近似的值。具体来说，通过从状态 x_k 出发，运行一个称为基本启发式（*base heuristic*）或基本策略（*base policy*）的启发式控制方案，然后通过计算足够多的阶段内产生的费用总和来获得近似值 $\tilde{J}_k(x_k)$。[14]

[14]在译本中，我们把 “rollout” 这一方法翻译为 “策略前展”，意指其将基本策略作用下的系统轨迹以仿真的形式 “向前展开”。在仿真基础上通过前瞻优化得到的 “rollout policy” 我们则称为 “前展策略”。——译者注

策略前展的主要理论特性是费用改进（*cost improvement*）：使用某个基本启发式方法进行策略前展所获得的费用小于或等于该基本启发式方法所对应的费用。只要基本启发式方法满足一些简单条件（将在第 2 章讨论），这一结论对任何初始状态均成立。[15]

此外，策略前展方法还有几种变体，包括涉及多个启发式方法的版本、与其他形式的值空间近似方法相结合的版本、多步前瞻以及考虑随机不确定性的版本。我们将在后面讨论这些变体。目前，我们先关注一个具有有限控制数目的确定性动态规划问题。给定时刻 k 的状态 x_k，该算法会考虑从每个可能的下一个状态 x_{k+1} 开始的所有尾部子问题，并通过使用某种被称为“基本启发式”的算法对它们进行次优求解。

因此，当处于状态 x_k 时，策略前展方法在线生成所有 $u_k \in U_k(x_k)$ 所对应的下一状态 x_{k+1}，并利用基本启发式方法计算状态序列 $\{x_{k+1}, \ldots, x_N\}$ 以及控制序列 $\{u_{k+1}, \ldots, u_{N-1}\}$，使得

$$x_{t+1} = f_t(x_t, u_t), \quad t = k, \ldots, N-1,$$

以及对应的成本为

$$H_{k+1}(x_{k+1}) = g_{k+1}(x_{k+1}, u_{k+1}) + \cdots + g_{N-1}(x_{N-1}, u_{N-1}) + g_N(x_N).$$

随后，策略前展算法在所有 $u_k \in U_k(x_k)$ 中选择使得从阶段 k 到 N 的尾部费用表达式

$$g_k(x_k, u_k) + H_{k+1}(x_{k+1})$$

达到最小值的控制。

等价且更简洁地讲，策略前展算法在状态 x_k 下采用的控制 $\tilde{\mu}_k(x_k)$ 由下面的最小化确定：

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k), \tag{1.12}$$

其中，近似 Q-因子 $\tilde{Q}_k(x_k, u_k)$ 定义为

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + H_{k+1}(x_{k+1}), \tag{1.13}$$

参见图 1.2.8。

请注意，策略前展算法需要运行基本启发式的次数最多为 Nn，其中 n 是每个状态可用控制选择数的上界。因此，如果 n 相对于 N 较小，则所需计算量相当于基本启发式单次应用计算时间的一个小倍数乘以 N。类似地，如果 n 被 N 的多项式所界定，则策略前展算法的计算时间与基本启发式计算时间之比也是 N 的一个多项式。

例 1.2.3 (旅行商问题-续)**.** 现在让我们考虑旅行商问题。给定 N 座城市 $c = 0, \ldots, N-1$，一位商人想要找出途径所有城市一次后返回出发城市的最短/花费最少的路径（参见例*1.2.2*）。对于每对不同的城市 c、c'，我们给定旅行费用 $g(c, c')$。注意到此处我们假设从任意城市出发都可以直接到达另一城市。该假设并不使问题失去一般性，这是因为我们可以将实际没有道路直接相连的城市对 (c, c') 的费用 $g(c, c')$ 设的极高。问题即为找出到访各城市的顺序，从而保证每个城市均被访问一次且总费用最小。

[15] 我们在此给出费用改进的直观说明。注意到策略前展对应的控制 $\tilde{u}_k$ 是根据公式(1.11)计算出来的，其目标是使得 u_k 取值使下面两项之和达到最小值：第一阶段的费用 $g_k(\tilde{x}_k, u_k)$ 加上采用基本启发式从第 $k+1$ 阶段到 N 阶段的费用。因此，策略前展不是直接采用基本启发式，而是在第一阶段进行了优化，这正是费用改进的依据。这种推理也解释了为何在策略前展方法中，多步前瞻通常比一步前瞻能提供更好的性能。

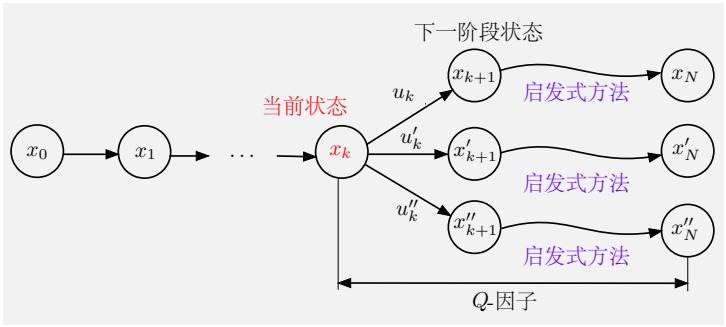


图 1.2.8: 确定性问题中一步前瞻策略前展的示意图：在状态 x_k 下，对于每个属于 $U_k(x_k)$ 的控制 u_k[即每个状态控制对 (x_k, u_k)]，基本启发式方法生成一个近似 Q-因子

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + H_{k+1}(x_{k+1}),$$

并选择使 Q-因子最小的控制 $\tilde{\mu}_k(x_k)$。

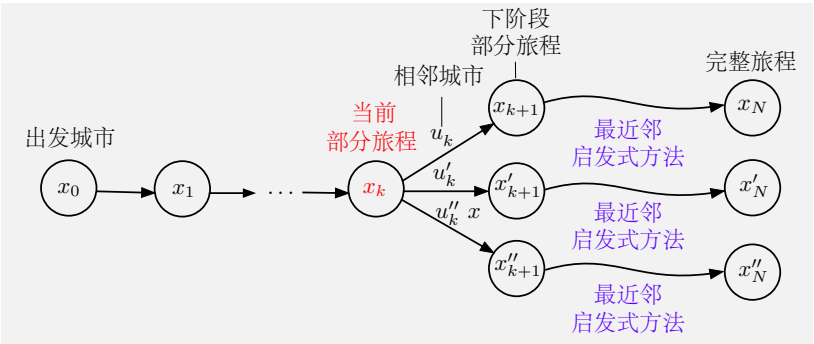
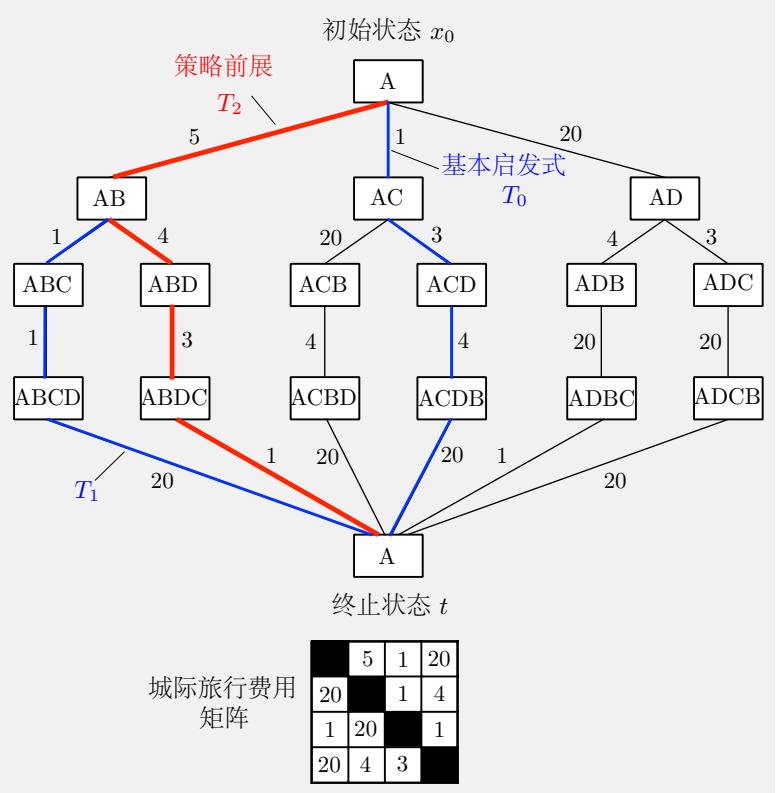


图 1.2.9: 采用策略前展求解旅行商问题图示。其中的基本启发式方法为最近邻启发式方法。初始状态 x_0 仅含有一座城市，而最后的状态 x_N 则对应于含有 N 座城市的完整旅程，其中每座城市的访问次数有且仅有一次。

有许多启发式方法可用于求解旅行商问题。为了便于讲解，我们着眼于简单的最近邻 (nearest neighbor) 启发式方法。给定一个部分旅程，即由几座不同城市构成的序列，该启发式方法在序列的末尾添加一座城市，同时确保新加的城市不与已有的城市序列构成环，且使访问新城市的成本最低。具体而言，给定由不同城市构成的序列 $\{c_0, c_1, \ldots, c_k\}$，最近邻方法在所有 $c_{k+1} \neq c_0, c_1, \ldots, c_k$ 中，选取最小化 $g(x_k, x_{k+1})$ 的城市 c_{k+1}，从而得到序列 $\{c_0, c_1, \ldots, c_k, c_{k+1}\}$。以此类推，该方法最终将构造出由 N 座城市 $\{c_0, c_1, \ldots, c_{N-1}\}$，从而获得一个完整旅程，且总费用为

$$g(c_0, c_1) + \cdots + g(c_{N-2}, c_{N-1}) + g(c_{N-1}, c_0). \tag{1.14}$$

正如例*1.2.2*中所述，我们可以将旅行商问题表述为动态规划问题。首先选取出发城市，即 c_0，作为初始状态 x_0。每个状态 x_k 对应于由不同的城市构成的部分旅程 $(c_0, c_1, \ldots, c_k)$。那么 x_k 的后续状态 x_{k+1} 就是形如 $(c_0, c_1, \ldots, c_k, c_{k+1})$ 的序列。这些序列是在已有的部分旅程的最后添加未访问的城市 $c_{k+1} \neq c_0, c_1, \ldots, c_k$ 而得到的（因此这些尚未访问的城市就构成了给定部分旅程/状态的可行控制集）。终止状态是形如 $(c_0, c_1, \ldots, c_{N-1}, c_0)$ 的完整旅程，且沿途选择后续城市所花费用的总和就是式(1.14)给出的完整旅程费用。



	5	1	20
20		1	4
1	20		1
20	4	3	

图 1.2.10: 使用最近邻基本启发式的策略前展算法在旅行商问题中的示意图。在城市 A，最近邻启发式方法生成了完整路线 ACDBA（记作 T_0）。在城市 A，策略前展算法比较了完整路线 ABCDA、ACDBA 和 ADCBA，发现费用最低的是 ABCDA（记作 T_1），于是转移到城市 B。在状态 AB 下，策略前展算法比较了完整路线 ABCDA 和 ABDCA，发现费用最低的是 ABDCA（记作 T_2），于是转移到城市 D。随后，策略前展算法依次前往城市 C 和 A（因无其他选择）。在此例中，策略前展算法生成的最终完整路线 T_2 证明是最优的，而基本启发式方法生成的完整路线 T_0 则是次优的。这预示着一个普遍的结论：对于确定性问题，只要基本启发式方法满足某些条件（这些条件将在第 2 章讨论，并且最近邻启发式满足这些条件），策略前展算法就能生成一系列费用逐步降低的解。

现在让我们用最近邻方法作为基本启发式方法。相应的策略前展方法由以下步骤组成：在 $k < N-1$ 次迭代之后，我们处于状态 x_k，即由不同城市构成的序列 $\{c_0, c_1, \ldots, c_k\}$。在下一步迭代中，我们以所有形如 $\{c_0, c_1, \ldots, c_k, c\}$，且 $c \neq c_0, c_1, \ldots, c_k$ 的序列为起点，运行最近邻启发式方法来得到完整旅程。然后我们选择通过最近邻启发式方法所花费用最小的城市 c 作为下一个城市 c_{k+1}；参见图 *1.2.9*。策略前展解的总体计算量以 N 的多项式为界，并且远小于精确动态规划的计算量。图*1.2.10*给出了一个例子，在该例中比较了最近邻启发式方法与相应的策略前展算法；另见习题 *1.1*。

1.3 随机动态规划——强化学习

现在我们将把动态规划算法及关于值空间近似的讨论扩展到那些在系统方程和费用函数中涉及随机不确定性的问题。我们首先讨论有限阶段情形，以及最优性原理和值空间近似方法所蕴含思想在随机问题中的应用。接着，我们将讨论无穷阶段问题，并概述相应的基

本理论和算法方法。

1.3.1 有限阶段问题

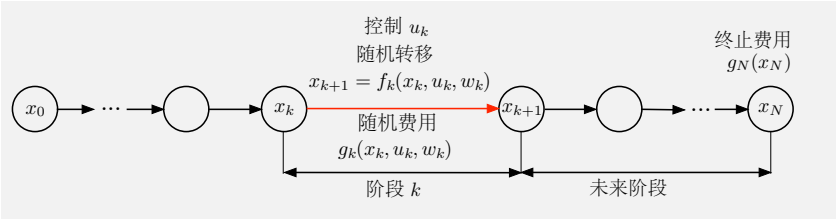


图 1.3.1: 一个 N 阶段的随机最优控制问题。从 x_k 出发，在控制信号 u_k 的作用下，下一时刻的状态依据以下系统方程 $x_{k+1} = f_k(x_k, u_k, w_k)$ 随机生成，其中 w_k 是随机扰动，并产生一个取值为 $g_k(x_k, u_k, w_k)$ 的随机阶段费用。

为了描述系统状态 x_k 的更新，随机有限阶段最优控制问题同样包含离散时间的动态系统。但与确定性问题相比，随机问题涉及的动态系统表达式中包括了一个随机“扰动” w_k。我们把扰动 w_k 服从的概率分布记作 $P_k(\cdot \,|\, x_k, u_k)$。该分布可能显式地由 x_k 与 u_k 的取值决定，但却不受先前扰动 $w_{k-1}, \ldots, w_0$ 取值的影响。该系统可写作如下形式

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N-1,$$

其中，与确定性问题一样，x_k 是某状态空间 S_k 的一个元素，控制 u_k 是某控制空间的一个元素。[16]每阶段的费用记作 $g_k(x_k, u_k, w_k)$，且费用取值也依赖于随机扰动 w_k；见图1.3.1。对于给定的当前状态 x_k，存在一个给定的、随状态变化的子集 $U_k(x_k)$，而控制 u_k 属于给定的约束集合 $U_k(x_k)$。

给定一个初始状态 x_0 和一个策略 $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$，未来状态 x_k 和扰动 w_k 都是随机变量。它们的概率分布由系统方程间接给出

$$x_{k+1} = f_k\big(x_k, \mu_k(x_k), w_k\big), \quad k = 0, 1, \ldots, N-1.$$

因此对于给定的阶段费用函数 g_k, $k = 0, 1, \ldots, N$，初始状态为 x_0 时策略 π 的期望费用为

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right\},$$

其中期望值运算 $E\{\cdot\}$ 是针对所有的随机变量 x_k 和 w_k。[17]一个最优策略 π^* 即为能够使上述费用最小化的策略，即

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0),$$

其中 Π 为所有策略构成的集合。

[16]离散方程格式以及相应的 $x - u - w$ 符号表示是最优控制文献中的标准格式。对于有限状态随机问题，也称为马尔可夫决策问题（*Markovian Decision Problems*，英文简写作 *MDP*），采用依赖于控制的转移概率来表示系统更为方便。强化学习文献中常见的符号是 $p(s, a, s')$，表示在采取动作 a 后，从状态 s 转移到状态 s' 的概率。这种符号表示不太适用于确定性问题，因为确定性问题根本不涉及概率结构，而确定性问题正是本书重点讨论的内容。此外，对于具有连续状态空间的问题，转移概率符号也显得非常繁琐；有关更多讨论，请参见第 1.7.1 节和第 1.7.2 节。然而，读者应注意，从数学角度来看，系统方程和转移概率是等价的，任何在一种符号系统下可以进行的分析，都可以转换到另一种符号系统下进行。

[17]我们假定读者具备概率论的入门知识。对于与本书中概率使用一致的介绍，请参阅 Bertsekas 和 Tsitsiklis 的教科书 [BT08]。

在确定性问题中，我们从所有的控制序列 $\{u_0,\dots,u_{N-1}\}$ 中找寻最优解，[参见式(1.3)]。与之不同的是，在随机问题中，我们从策略（*policies*）（也称为闭环控制律（*closed-loop control laws*）或反馈策略（*feedback policies*））中选取最优解。每个策略是由一个函数序列构成

$$\pi=\{\mu_0,\dots,\mu_{N-1}\},$$

其中 μ_k 是从状态 x_k 到控制 $u_k=\mu_k(x_k)$ 的映射，并且满足控制约束，即对所有 x_k，约束 $\mu_k(x_k)\in U_k(x_k)$ 成立。相较于控制序列，策略是更一般的数学概念。因为策略允许基于当前状态 x_k 提供的信息来选择控制 u_k，所以在存在随机不确定性的情况下，策略可以降低所需的费用。如果没有当前状态提供的信息，当状态的取值并非预期时，控制器就不能充分应对，从而给所花费的费用带来不利影响。以上所述是确定性与随机最优控制问题的本质区别。

最优费用取决于 x_0 并记作 $J^*(x_0)$，即

$$J^*(x_0)=\min_{\pi\in\Pi}J_\pi(x_0).$$

为了便于理解，我们可以把 J^* 视作给每个初始状态 x_0 赋值最优费用 $J^*(x_0)$ 的函数，并称之为最优费用函数（*optimal cost function*）或最优价值函数（*optimal value function*）。

随机动态规划

与针对确定性问题的动态规划算法相比，适用于随机有限阶段最优控制问题的动态规划算法具有相似的形式，并且具有与确定性版本相同的几个主要特征

(a) 通过利用尾部子问题，将多阶段的最小化问题分解为单阶段的最小化问题。

(b) 对所有的阶段 k 和状态 x_k，从后向前求解相应的 $J_k^*(x_k)$ 的取值，即从 k 时刻的状态 x_k 出发所产生的最优展望费用。

(c) 通过在动态规划公式中的最小化运算来获得一个最优控制策略。

(d) 适于采用值空间近似的结构。在该结构中，我们用近似的 $\tilde{J}_k$ 来代替 J_k^*，并通过相应的最小化求解来获得一个次优解。

针对随机有限阶段问题的动态规划算法. 首先令方程

$$J_N^*(x_N)=g_N(x_N).$$

对 $k=0,\dots,N-1$，令

$$J_k^*(x_k)=\min_{u_k\in U_k(x_k)}E\Big\{g_k(x_k,u_k,w_k)+J_{k+1}^*\big(f_k(x_k,u_k,w_k)\big)\Big\}\tag{1.15}$$

成立。如果对于每一个状态 x_k 及时刻 k，上式右侧在 $u^*=\mu_k^*(x_k)$ 取得最小值，那么策略 $\pi^*=\{\mu_0^*,\dots,\mu_{N-1}^*\}$ 则为最优策略。

关键的事实是，从任何初始状态 x_0 出发，最优费用等于上述动态规划算法最后一步得到的数值 $J_0^*(x_0)$。这一点可以通过归纳法证明，证明方法类似于确定性情况；这里我们将省

略证明（顺便提一下，该证明涉及一些数学细节；有关讨论请参见教科书 [Ber17] 的第 1.3 节）。

在对最优展望费用函数 $J_0^*, \dots, J_N^*$ 进行离线计算的同时，我们可以通过在公式(1.15)中进行最小化来计算和存储一个最优策略 $\pi^* = \{\mu_0^*, \dots, \mu_{N\to 1}^*\}$。然后，我们可以在在线控制过程中使用这个策略，即在到达状态 x_k 时，从存储中取出并应用控制 $\mu_k^*(x_k)$。另一种选择是放弃存储策略 π^*，而是通过在线执行最小化(1.15)来计算控制 $\mu_k^*(x_k)$。

在少数有利的情况下，可以利用随机动态规划算法解析地计算出最优展望费用函数 J_k^* 和最优策略 μ_k^*。其中一个知名的例子涉及线性系统和二次型费用函数，这是控制理论中的一类基本问题。接下来，我们将展示这一问题的标量版本。类似的分析可以推广到多维系统（参见如 [Ber17] 等最优控制教材）。

例 1.3.1 (线性二次型最优控制)**.** 在此问题中，系统是线性的，

$$x_{k+1} = ax_k + bu_k + w_k, \qquad k = 0, \dots, N-1,$$

且状态、控制和扰动均为标量。此外，随机扰动 w_k 是具有零均值和给定方差 σ^2。费用为二次型：

$$qx_N^2 + \sum_{k=0}^{N-1} (qx_k^2 + ru_k^2),$$

其中 q 和 r 为已知的正的权重参数。我们假设 x_k 和 u_k 上没有任何约束（实际上，这类问题一般都包含约束，但我们可以先忽略这些约束，随后再检查是否存在严重违反约束的情况）。

举例来说，考虑一个在直线道路上行驶的车辆，该车辆受控制力 u_k 的作用且无摩擦。我们的目标是使车辆的速度保持在恒定水平 $\bar{v}$（类似于一个过于简化的巡航控制系统）。将其牛顿动力学进行时间离散化并加入随机噪声后，时刻 k 的速度 v_k 按下式演化：

$$v_{k+1} = v_k + bu_k + w_k. \tag{1.16}$$

通过引入 $x_k = v_k - \bar{v}$，即车辆在时刻 k 的速度 v_k 与目标速度 $\bar{v}$ 之间的偏差，我们得到系统方程

$$x_{k+1} = x_k + bu_k + w_k.$$

这里，系数 b 与车辆重量、道路状况等多个问题特性有关。费用函数则表达了我们希望以相对较小的控制力使 x_k 保持接近于零的愿望。

我们将应用动态规划算法，推导出最优展望费用函数 J_k^* 以及最优策略。首先已知

$$J_N^*(x_N) = qx_N^2,$$

然后根据公式(1.15)，我们得到

$$\begin{aligned}
J_{N-1}^*(x_{N-1}) &= \min_{u_{N-1}} E\big\{qx_{N-1}^2 + ru_{N-1}^2 + J_N^*(ax_{N-1} + bu_{N-1} + w_{N-1})\big\} \\
&= \min_{u_{N-1}} E\big\{qx_{N-1}^2 + ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1} + w_{N-1})^2\big\} \\
&= \min_{u_{N-1}} \big[qx_{N-1}^2 + ru_{N-1}^2 + q(ax_{N-1} + bu_{N-1})^2 \\
&\qquad + 2qE\{w_{N-1}\}(ax_{N-1} + bu_{N-1}) + qE\{w_{N-1}^2\}\}\big],
\end{aligned}$$

最后，利用假设条件 $E\{w_{N-1}\}=0$ 和 $E\{w_{N-1}^2\}=\sigma^2$，并将不依赖于的项移到最小化运算之外，我们就得到

$$J_{N-1}^*(x_{N-1})=qx_{N-1}^2+q\sigma^2+\min_{u_{N-1}}\left[ru_{N-1}^2+q(ax_{N-1}+bu_{N-1})^2\right]. \tag{1.17}$$

前面方程中对 u_{N-1} 进行最小化的表达式是关于 u_{N-1} 的凸二次函数，因此通过对 u_{N-1} 求导并设其为零：

$$0=2ru_{N-1}+2qb(ax_{N-1}+bu_{N-1}),$$

我们得到最后阶段的最优策略：

$$\mu_{N-1}^*(x_{N-1})=-\frac{abq}{r+b^2q}x_{N-1}.$$

将该表达式代入公式 *(1.17)*，通过简单的计算，我们得到

$$J_{N-1}^*(x_{N-1})=K_{N-1}x_{N-1}^2+q\sigma^2,$$

其中

$$K_{N-1}=\frac{a^2rq}{r+b^2q}+q.$$

我们现在可以继续动态规划算法，从 J_{N-1}^* 推导出 J_{N-2}^*。一个重要的观察是，J_{N-1}^* 是二次形式的（加上一个无关紧要的常数项），因此通过类似的计算，我们可以解析地求得 μ_{N-2}^* 和 J_{N-2}^*。，它们分别是 x_{N-2} 的线性函数和二次函数（加常数项）。这一过程可以一直向后进行，并且通过归纳法可以验证，对于所有 k，我们得到的最优策略和最优展望费用函数均具有以下形式：

$$\mu_k^*(x_k)=L_kx_k,\quad k=0,1,\dots,N-1,$$

$$J_k^*(x_k)=K_kx_k^2+\sigma^2\sum_{t=k}^{N-1}K_{t+1},\quad k=0,1,\dots,N-1,$$

其中

$$L_k=-\frac{abK_{k+1}}{r+b^2K_{k+1}},\quad k=0,1,\dots,N-1, \tag{1.18}$$

并且序列 $\{K_k\}$ 则由以下递推关系从后向前生成：

$$K_k=\frac{a^2rK_{k+1}}{r+b^2K_{k+1}}+q,\quad k=0,1,\dots,N-1, \tag{1.19}$$

其中起始终止条件为 $K_N=q$。

在本示例中，我们获得解析解的过程值得注意。回顾算法的各个步骤，读者会发现使得解法简化的关键在于费用函数的二次性质和系统方程的线性属性。事实上，可以一般性地证明，当系统是线性且阶段费用为二次时，即使对于多维线性系统，最优策略和最优展望费用函数也可以用解析表达式给出（参见 *[Ber17]* 第 *3.1* 节）。最优策略是状态的线性函数，而最优费用函数则是状态的二次函数加上一个常数。

这个例子的另一个显著特点（同样可推广到多维系统）是：最优策略不依赖于 w_k 的方差，并且当 w_k 被其均值（在本例中为零）替换时，最优策略仍然不受影响。这被称为确定性等价（certainty equivalence），并且出现在涉及线性系统和二次费用的多种问题中；参见

[Ber17] 第 3.1 节和第 4.2 节。例如，即使 w_k 具有非零均值，该性质仍然成立。对于其他问题，确定性等价可作为问题近似的基础，例如，假设确定性等价成立（即将随机量替换为一些典型值，如它们的期望值），然后对所得的确定性最优控制问题应用精确动态规划。这是强化学习方法论的重要部分，我们将在本章后续讨论，并在第 2 章中更详细地介绍。

请注意，上述例子中所示的线性二次型问题是个例外，因为它可以得到解析解。而在实际应用中遇到的大多数动态规划问题都需要依赖数值计算方法来求解。

随机问题的 Q-因子和 Q-学习

类似于确定性问题中的定义方式 [见式(1.9)]，我们可以定义随机问题的最优 Q-因子，即将随机动态规划等式(1.15)右侧中进行最小化运算的部分定义为最优 Q-因子。由此可知最优 Q-因子如下

$$Q_k^*(x_k,u_k)=E\Big\{g_k(x_k,u_k,w_k)+J_{k+1}^*\big(f_k(x_k,u_k,w_k)\big)\Big\}. \tag{1.20}$$

最优展望费用函数 J_k^* 则可通过对最优 Q-因子作如下运算得到

$$J_k^*(x_k)=\min_{u_k\in U_k(x_k)}Q_k^*(x_k,u_k),$$

而且动态规划算法可以表述为只涉及 Q-因子的如下形式

$$Q_k^*(x_k,u_k)=E\left\{g_k(x_k,u_k,w_k)+\min_{u_{k+1}\in U_{k+1}(f_k(x_k,u_k,w_k))}Q_{k+1}^*\big(f_k(x_k,u_k,w_k),u_{k+1}\big)\right\}.$$

稍后我们将关注近似 Q-因子，其中公式(1.20)中的 J_{k+1}^* 被替换为近似函数 $\tilde{J}_{k+1}$。同样，与状态控制对 (x_k,u_k) 对应的 Q-因子是使用该状态-控制对产生的第一阶段费用的期望，加上从下一状态开始的剩余阶段费用的期望（后者由函数 $\tilde{J}_{k+1}$ 估计）之和。

1.3.2 随机动态规划问题的值空间近似

通常，最优展望费用函数 J_k^* 的计算可能非常耗时，甚至不可能实现。为应对这一困难，强化学习的主要方法之一是在值空间中进行近似。在这种方法中，使用近似函数 $\tilde{J}_k$ 来代替 J_k^*，与确定性情况类似；参见公式(1.8)和(1.11)。

值空间的近似——用 $\tilde{J}_k$ 代替 J_k^*. *在 k 时刻的任意状态 x_k，计算并采用如下决策*

$$\tilde{\mu}_k(x_k)\in\arg\min_{u_k\in U_k(x_k)}E\Big\{g_k(x_k,u_k,w_k)+\tilde{J}_{k+1}\big(f_k(x_k,u_k,w_k)\big)\Big\}. \tag{1.21}$$

式(1.21)中的一步前瞻最小化只需要针对系统在线控制过程中遇到的 N 个状态 $x_0,\ldots,x_{N-1}$ 进行。相比之下，精确的动态规划要求对每个状态和每个阶段都进行此类最小化。

三种近似

在设计值空间近似方法时，可以考虑几种有趣的简化思路，旨在减轻计算负担。除了费用函数近似（即使用 $\tilde{J}_{k+1}$ 代替 J_{k+1}^*）之外，还有其他可能性。其中之一是，根据某种启

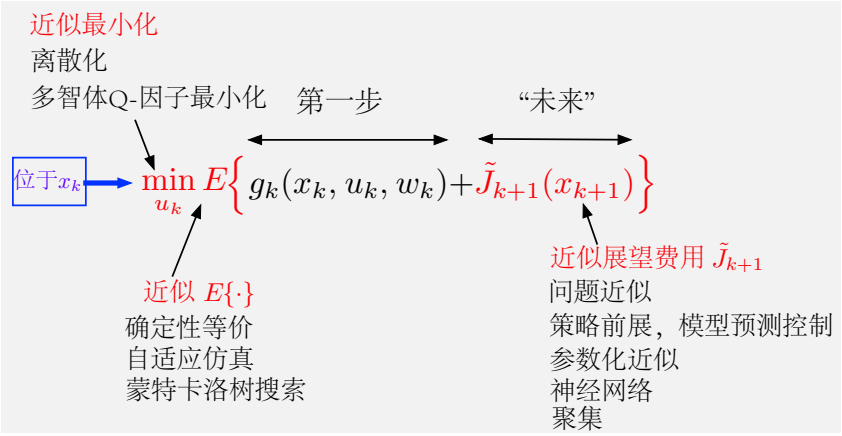


图 1.3.2: 随机问题中值空间近似的示意图，以及其设计中涉及的三种近似。通常，这些近似方法可以彼此独立地设计，并且可采用多种方法。此外，还存在多步前瞻版本的值空间近似方法，稍后将进行讨论。

发式标准构造一个控制约束集 $U_k(x_k)$ 的子集，并用该子集代替原控制约束集，从而简化在 $u_k \in U_k(x_k)$ 上的前瞻最小化 [参见式(1.15)]。

在第 2.9 节中，我们将讨论一个相关的思路，用于多智能体情况下的控制空间简化。在此类问题中，控制由多个组分构成，即 $u_k = (u_k^1, \ldots, u_k^m)$。此时，可以依次对每个组分进行单独的最小化，从而可能显著降低计算量。

另一种简化方法是，通过使用有限的蒙特卡罗仿真，对公式(1.21)中期望值的计算进行近似。蒙特卡罗树搜索方法（将在第 4 章的第 4.1.5 节中讲解）就是这一类型方法的一种可能方案。

另一种期望值简化的方法则是基于确定性等价方法（*certainty equivalence approach*），我们将在第 4 章的第 4.1.2 节中详细讨论此类方法。在这种方法中，在第 k 阶段，我们将未来的随机变量 $w_{k+1}, \ldots, w_{k+m}$ 替换为某些确定性值 $\overline{w}_{k+1}, \ldots, \overline{w}_{k+m}$，例如它们的期望值。我们也可以将这种做法视为一种问题近似方法，即为了计算 $\tilde{J}_{k+1}(x_{k+1})$，我们“假设”问题是确定性的，将未来的随机量替换为确定性的典型值。这是使得随机问题中的值空间近似方法在计算上可行的最有效的技术之一。该方法的这一优势在它与多步前瞻最小化相结合时尤为明显（我们将在后面讨论）。

图1.3.2展示了在随机问题的值空间近似中涉及的三种近似：展望费用近似（*cost-to-go approximation*）、简化最小化（*simplified minimization*）以及期望值近似（*expected value approximation*）。它们在很大程度上可以相互独立地设计，并可通过多种方法实现。本书的大部分讨论将围绕如何在一步前瞻和多步前瞻两种情形下设计这三种近似方法展开。

如图1.3.2所示，对于展望费用近似，一个重要的方法是问题近似（*problem approximation*）。在此类方法中，我们构造一个与原问题相关的简化的优化问题。该问题应当更便于求解，而其最优或近似最优的费用函数则可用作式(1.21)中的函数 $\tilde{J}_{k+1}$。这里的简化方法可能包括利用可分解结构、忽略各种不确定性以及减少状态空间的规模。作者的强化学习著作 [Ber19b] 中讨论了几种问题近似的方法。其中一种主要方法是聚集（*aggregation*），我们将在第 4.6 节中讨论。尽管问题近似常常可以与我们主要关注的值空间近似方法非常有效地结合，但在本书中，问题近似将不会受到过多关注。

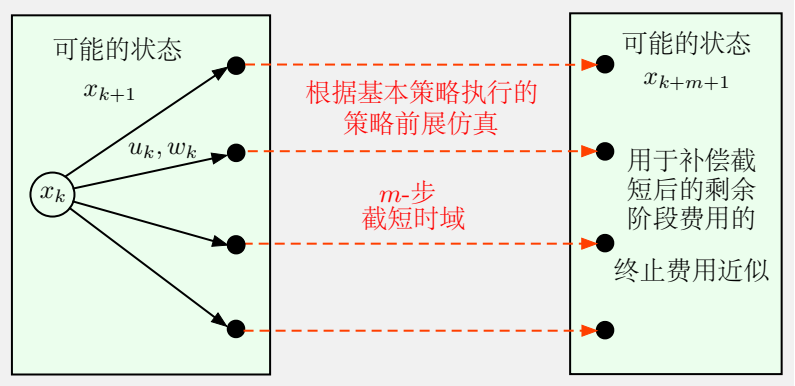


图 1.3.3: 截短策略前展的示意图。先进行一步前瞻，然后对基本策略进行 m 步仿真，并将一个近似费用 $\tilde{J}_{k+m+1}(x_{k+m+1})$ 加入到仿真费用中，该费用依赖于前展仿真结束时获得的状态 x_{k+m+1}。如果省略基本策略仿真（即 $m=0$），则恢复了值空间中一般的近似方案(1.21)。同样，也可以采用多步前瞻的截短策略仿真，我们将在第 2 章中详细讨论此类方法。

另一种用于在线展望费用函数近似的重要方法是策略前展，我们将在接下来讨论。这与第 1.2 节中讨论的确定性问题的策略前展方法类似。

随机问题的策略前展——截短策略前展

在策略前展方法中，我们将某个合适的基本策略的费用函数（或者该函数的近似）用作公式 (1.21) 中的 $\tilde{J}_{k+1}$。值得注意的是，任何策略都可以在线上用作基本策略，包括那些通过复杂的离线方法获得的策略，例如利用神经网络和训练数据得到的策略。策略前展算法具有费用改进的特性，即相对于其基本策略，它能够获得改进后的费用。我们将在第 2 章中讨论这一特性以及保证其成立的一些条件。

策略前展的一个主要变种是截短策略前展（*truncated rollout*）。该方法将一步优化和对基本策略进行的多步仿真相结合，并将一个近似费用 $\tilde{J}_{k+m+1}(x_{k+m+1})$ 加入到仿真费用中。此处的费用将依赖于策略前展结束时获得的状态 x_{k+m+1}。请注意，如果放弃使用基本策略（即 $m=0$），那么我们就得到了值空间中一般近似方案(1.21)；参见图1.3.3。因此，策略前展为一步最小化提供了额外的前瞻层次，但这种前瞻不必延伸到整个决策时域的末尾。

还应注意，截短策略前展也有多步前瞻最小化的版本，我们将在后文中讨论该方法。在无穷阶段问题中，由于无法进行无限步的基本策略的前展仿真，终止费用近似显然是必要的。然而，即使在有限阶段问题中，人为地截短策略前展的时域也可能是必要和/或有益的。通常，多步前瞻最小化与策略前展的步数的之和较大往往是有利的。

费用与 Q-因子近似——鲁棒性与在线重新规划

类似于确定性情况，Q-学习涉及计算最优 Q-因子(1.20)或其近似值 $\tilde{Q}_k(x_k,u_k)$。这些近似 Q-因子可以通过值空间近似方法获得。然后我们可以通过最小化 Q-因子来得到最优或近似最优的策略，即

$$\tilde{\mu}_k(x_k) \in \arg\min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k,u_k). \tag{1.22}$$

由于可以通过使用费用函数近似 [参见式(1.21)] 或 Q-因子近似 [参见式(1.22)] 来实现

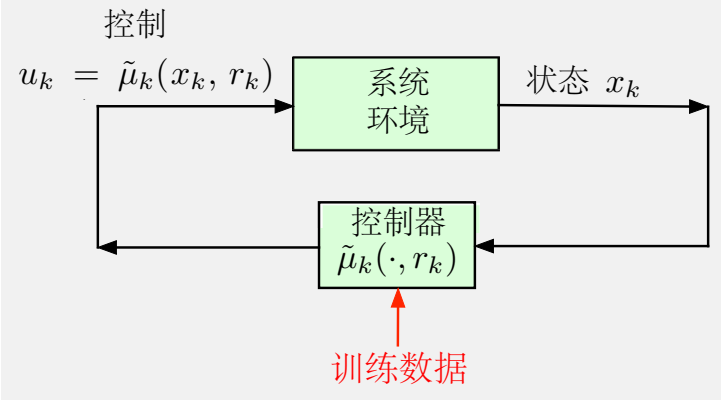


图 1.3.4: 参数化策略空间近似的示意图。该方法基于数据离线计算得到一个属于某种参数化类别的策略

$$\tilde{\mu}_k(x_k, r_k), \qquad k = 0, \ldots, N-1.$$

在状态 x_k 时，该策略用于在线生成控制 $u_k = \tilde{\mu}_k(x_k, r_k)$。

值空间近似，因此在实际应用中该使用哪一种方法便成了一个问题。一个重要的考虑因素是获得合适的费用函数或 Q-因子近似的难易程度，这在很大程度上取决于具体问题以及可用于构建近似函数的数据信息的可用性。然而，还有一些其他主要的考虑因素。

具体来说，费用函数近似方案

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} E\Big\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big)\Big\}. \tag{1.23}$$

有一个重要缺点：对所有 $u_k \in U_k(x_k)$，我们需要在线计算相应的期望值，这可能涉及大量的计算工作（*the expected value above needs to be computed on-line for all* $u_k \in U_k(x_k)$*, and this may involve substantial computation*）。它也有一个重要优点，即在系统运行过程中，如果系统函数 f_k、每阶段费用 g_k 或控制约束集 $U_k(x_k)$ 发生变化，假设新的 f_k、g_k 或 $U_k(x_k)$ 在时刻 k 被控制器获取，那么可以使用在线重新规划，这可能大大提高值空间近似方案的鲁棒性（*on-line replanning may be used, and this may improve substantially the robustness of the approximation in value space scheme*）。相比之下，Q-因子函数近似方案(1.22)不支持在线重新规划。另一方面，对于那些不需要在线重新规划的问题，Q-因子近似方案可能不需要在线计算期望值，并且能够通过式(1.22)更快地在线求解出最小化控制 $\tilde{\mu}_k(x_k)$。

后续当我们讨论基于牛顿法的离线训练与在线执行之间的协同效应时（参见第 1.5 节），我们还将提到使用 Q-因子方法的另一个缺点。具体来说，我们将把前瞻最小化所得的策略 $\{\tilde{\mu}_0, \ldots, \tilde{\mu}_{N-1}\}$ 的费用函数给出新的解释，即将其视为从终止费用函数近似 $\{\tilde{J}_1, \ldots, \tilde{J}_N\}$ 出发，通过牛顿法求解动态规划问题对应的贝尔曼方程的单步迭代后得到的结果。当使用 Q-因子近似（而非费用函数近似）时，这种协同效应往往会受到负面影响。

1.3.3 策略空间近似

值空间近似的主要替代方法是策略空间近似（*approximation in policy space*），即我们从一个适当受限的策略集合中选择策略，通常是某种形式的参数化集合。具体来说，我们可

以引入一个参数化策略族（即第 3 章所谓的近似架构）：

$$\tilde{\mu}_k(x_k, r_k), \qquad k = 0, \ldots, N-1,$$

其中 r_k 是参数，然后通过某种训练过程或优化方法来估计这些参数 r_k；参见图 1.3.4。

在本节以及整本书中，我们重点讨论离线选择策略的方法（*we focus on selecting a policy off-line*），其中可能涉及到利用离线收集的数据进行训练。也有一些算法旨在利用在线收集的数据来改进参数化的策略，但此类超出了本书的讨论范围（参见第 5 章中关于策略梯度方法的相关讨论）。

神经网络（将在第 5 章中描述）常用作策略近似中的参数化类。在这种情况下 r_k 就是神经网络的权重/参数向量。在第 5 章中，我们还将讨论获取用于确定参数 r_k 所需训练数据的方法，并考虑其他几种近似架构。

在策略空间中的参数化近似的一般方案是以某种方式获得一个训练集，该训练集由大量的状态-控制对样本 $(x_k^s, u_k^s), s = 1, \ldots, q$ 组成，且对于每个 s，u_k^s 都是在状态 x_k^s 下一个“良好”的控制。然后，我们可以通过求解最小二乘/回归问题来选择参数 r_k：

$$\min_{r_k} \sum_{s=1}^{q} \|u_k^s - \tilde{\mu}_k(x_k^s, r_k)\|^2 \tag{1.24}$$

（可能通过添加正则化进行修改）。[18]具体而言，我们可以通过一个人类或软件“专家”来确定 u_k^s，该专家能够在给定状态下选择“接近最优”的控制，因此 $\tilde{\mu}_k$ 被训练以匹配专家的行为。这类方法在人工智能中通常被称为监督学习（*supervised learning*）。

生成最小二乘训练问题(1.24)的训练集 $(x_k^s, u_k^s), s = 1, \ldots, q$ 的一个重要方法是基于值空间近似。具体来说，我们可以使用如下形式的一步前瞻最小化方法：

$$u_k^s \in \arg\min_{u \in U_k(x_k^s)} E\Big\{ g_k(x_k^s, u, w_k) + \tilde{J}_{k+1}\big(f_k(x_k^s, u, w_k)\big) \Big\},$$

其中 $\tilde{J}_{k+1}$ 是一个合适的（另行获得的）值空间近似。或者，我们也可以使用基于近似 Q-因子的最小化方法

$$u_k^s \in \arg\min_{u \in U_k(x_k^s)} \tilde{Q}_k(x_k^s, u),$$

其中 $\tilde{Q}_k$ 是一个（另行获得的）Q-因子近似。我们可以将这种方法视为值空间近似基础上构建的策略空间近似（*approximation in policy space built on top of approximation in value space*）。

式(1.24)中的最小二乘训练过程，以及更一般的策略空间近似，有一个显著优势：一旦得到参数化策略 $\tilde{\mu}_k$，在系统在线运行时计算控制

$$u_k = \tilde{\mu}_k(x_k, r_k), \qquad k = 0, \ldots, N-1,$$

[18]这里 $\|\cdot\|$ 表示标准的二次欧几里得范数。这里（以及后面类似的情况中）的隐含假设是控制属于欧几里得空间（即具有实值组分的有限维向量空间），因此可以通过它们的范数差来衡量两个控制之间的距离（随机控制，即某个特定动作被采用的概率，也属于这一范畴）。这类回归问题出现在基于数据训练参数化分类器（*parametric classifiers*）的过程中，其中就包括了应用神经网络的分类器（参见第 5.4 节）。在假定控制空间有限的情况下，分类器使用数据 $\{(x_k^s, u_k^s)\}_{s=1}^q$（被视为状态-类别对）进行训练，然后将一个状态 x_k 分类为“类别” $\tilde{\mu}_k(x_k, r_k)$。参数化近似架构以及通过分类和回归技术进行训练的方法将在第 5 章中详细介绍。一项重要的改进是采用正则化回归（*regularized regression*），即在最小二乘目标中加入一个二次正则化项，该项是 r 与某个初始猜测 $\hat{r}$ 之间偏差平方 $\|r - \hat{r}\|^2$ 的正倍数。

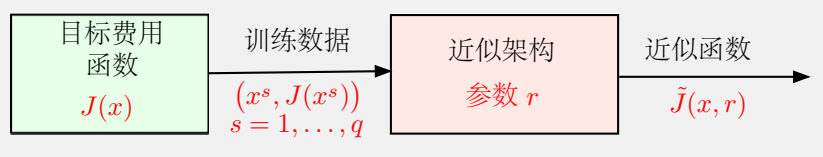


图 1.3.5: 参数化成本近似的一般结构。我们用参数化类别中的一个成员 $\tilde{J}(x,r)$ 来近似目标成本函数 $J(x)$，其中该成员依赖于参数向量 r。我们利用训练数据 $\big(x^s, J(x^s)\big)$，$s = 1, \ldots, q$，并采用一种优化方法，旨在找到一个参数 $\hat{r}$，使得对于 $s = 1, \ldots, q$，误差 $J(x^s) - \tilde{J}(x^s, \hat{r})$ 的取值“最小”。

通常比前瞻最小化(1.23)更为简单。因此，策略空间近似的主要用途之一是为已知策略提供一种近似实现（*approximate implementation of a known policy*）（无论该策略是如何获得的），便于在线使用。但从负面来看，这种实现不太适合用于在线重新规划。

无模型的策略空间近似

文献中还有一些基于优化的方法用于策略空间近似。其主要思路是，一旦我们使用向量 $(r_0, r_1, \ldots, r_{N-1})$ 来参数化策略 π，那么期望费用 $J_\pi(x_0)$ 也会被参数化，并且可以视为 $(r_0, r_1, \ldots, r_{N-1})$ 的函数。然后，我们可以使用类似梯度或随机搜索方法来优化该费用。这是在策略空间优化中广泛使用的一种方法，但在本书中将受到较少讨论（参见第 5.5 节，以及强化学习书籍 [Ber19b] 的第 5.7 节）。

这种方法的一个有趣特点在于，原则上它不需要系统和成本函数的数学模型；只需一个计算机仿真器（或可用于实验的真实系统）即可。这有时被称为无模型实现（*model-free implementation*）。采用这种方法是否明智，尤其是当其完全依赖于仿真（即不使用任何先前的数学模型知识）时，是一个备受争议和广泛讨论的问题；例如，参见 Alamir[Ala22] 的综述论文。[19]

最后，我们指出值空间近似与策略空间近似之间一个重要的概念区别。前者主要是一种在线方法（在一步或多步前瞻中，可选择性地使用离线训练来构造费用函数近似），而后者主要是一种离线训练方法（可直接用于在线运行，或可以为在线策略前展提供基本策略）。

1.3.4 近似费用函数和策略的离线训练

对于离线构造的近似，一个主要方法是使用参数化近似。在强化学习领域中，基于特征的架构和神经网络非常有用，这将在第 3 章中讨论这些架构，同时介绍用于训练它们的方法。[20]

[19] “无模型”这一术语可能会令人困惑。实际上，在动态规划/强化学习问题的表述中总是存在某种模型（*here is always a model in DP/RL problem formulations*），区别只在于它是数学模型（即基于方程）、计算机模型（即基于计算机仿真），还是混合模型（即同时依赖数学方程和仿真）。

[20] 在本书中，神经网络的主要作用是提供一种从输入-输出数据中近似各种目标函数的方法。这些目标函数包括给定策略的费用函数和 Q-因子，以及最优的展望费用函数和 Q-因子；在这种情况下，神经网络被称为价值网络（*velue network*）[有时也称为评论家网络（critic network）]。在其他情况下，神经网络代表一种策略，即将状态映射到控制，此时称为策略网络（*policy network*）(另一种说法是执行者网络（*actor network*))。从数据中构造费用函数、Q-因子和策略近似的训练方法主要基于优化和回归。我们将在第 3 章中进行讲解。在动态规划的视角下对这些方法的讨论可在许多文献中找到，包括强化学习书籍 [Ber19b]、[Ber20b] 以及神经元动态规划书籍 [BT96]。此外，感兴趣的读者还可以阅读机器学习书籍，其中详细描述了神经网络架构和训练方法，例如 Bishop 和 Bishop 的最新著作 [BB23] 及其引用的相关文献。

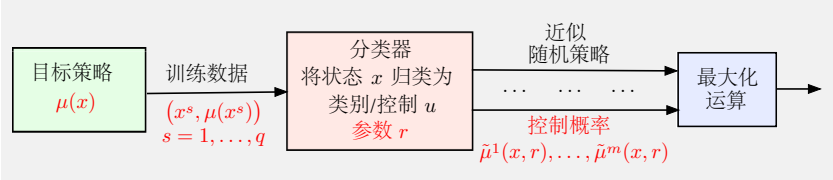


图 1.3.6: 有限控制空间 $U = \{u^1, \ldots, u^m\}$ 的情况下，参数化策略近似的一般结构及其与分类方案的关系。该方法产生一个形如(1.25)的随机策略，并通过最大化运算(1.26)将其转换为非随机策略。

参数化费用函数近似的一般结构如图1.3.5所示。我们有一个目标函数 $J(x)$，希望用一个依赖于参数向量 r 的参数化函数 $\tilde{J}(x,r)$ 来进行近似（为简化起见，我们省略了时间索引，使用 J 代替 J_k）。为此，我们收集训练数据 $\{(x^s, J(x^s))\}$，$s = 1, \ldots, q$，并利用这些数据来确定一个参数 $\hat{r}$，使得参数化函数的预测值 $\tilde{J}(x^s, \hat{r})$ 与相应数据 $J(x^s)$ 之间的“拟合”效果较好。确定参数 $\hat{r}$ 通常是通过某种形式的优化方法来完成，其目标是在某种意义上最小化误差 $J(x^s) - \tilde{J}(x^s, \hat{r})$ 的大小。

参数化费用函数近似的方法和思想也可以用于目标策略 μ 的近似，即采用来自参数化函数族的策略 $\tilde{\mu}(x,r)$ 来近似目标策略。例如，训练数据可以通过策略前展的计算过程获得，从而构造出既有价值网络又有策略网络、并将二者结合起来的永续的策略前展方案。然而，对策略近似和对费用函数近似有一个重要区别：近似的费用函数值 $\tilde{J}(x,r)$ 是实数，而近似策略的取值 $\tilde{\mu}(x,r)$ 是控制空间 U 的元素。因此，如果 U 由 m 维向量组成，那么 $\tilde{\mu}(x,r)$ 包含 m 个数值分量。在这种情况下，成本函数和策略的参数化近似问题是非常相似的，且都涉及连续空间的近似。

另一方面，当控制空间是有限的，即 $U = \{u^1, \ldots, u^m\}$ 时，情况就大不相同了。在这种情况下，对于任意状态 x，$\tilde{\mu}(x,r)$ 由 m 个可能控制 $u^1, \ldots, u^m$ 中的一个构成。这就与传统的分类方案建立了联系，其中对象 x 被归类为 $u^1, \ldots, u^m$ 中的某个类别，因此$\mu(x)$ 定义了 x 的类别，并可被视为一个分类器（$\mu(x)$ *defines the category of* x*, and can be viewed as a classifier*）。一些最著名的分类方案实际上会产生随机结果，即 x 与一个概率分布关联：

$$\{\tilde{\mu}(u^1, r), \ldots, \tilde{\mu}(u^m, r)\} \tag{1.25}$$

在我们的策略近似视角下，这就是一个随机化策略；参见图1.3.6。构造随机策略通常是为了算法实现的便利，因为许多优化方法（包括最小二乘回归）要求优化变量必须是连续的。在这种情况下，随机化策略(1.25)可以通过最大化运算转换为非随机化策略：将 x 与具有最大概率的控制关联（参见图1.3.6），即

$$\tilde{\mu}(x,r) \in \arg\min_{i=1,\ldots,m} \tilde{\mu}^i(x,r). \tag{1.26}$$

在策略空间近似中使用分类方法的问题将在第 5 章（第 5.4 节）中讨论。

1.4 无穷阶段问题——概述

现在我们将概述无穷阶段随机动态规划理论，并重点介绍与我们的强化学习/近似方法相关的方面。更严谨的数学论述见附录 A。我们主要处理无穷阶段随机问题，其目标是最

小化无穷阶段累积的总费用，即

$$J_\pi(x_0) = \lim_{N\to\infty} \underset{\substack{w_k\\ k=0,1,\dots}}{E} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}; \tag{1.27}$$

参见图1.4.1。此处的 $J_\pi(x_0)$ 表示对应于初始状态 x_0 和策略 $\pi = \{\mu_0, \mu_1, \dots\}$ 的费用，而 α 则是处于区间 $(0, 1]$ 中的标量常数。定义每阶段成本的函数 g 和系统方程

$$x_{k+1} = f(x_k, u_k, w_k)$$

在各阶段之间保持不变。随机扰动 $w_0, w_1, \dots$ 具有相同的条件概率分布 $P(\cdot \mid x_k, u_k)$。

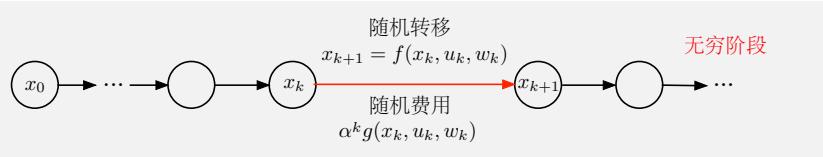


图 1.4.1: 无穷阶段问题图示。抛开所采用的折扣因子 α，系统和每阶段费用都是稳态的。如果 $\alpha = 1$，则问题中还会有一个我们希望达到的特殊的无费用终止状态。

如果 α 小于 1，那么它就具有折扣因子（*discount factor*）的含义，其作用是使未来的费用相对于当前发生的相同费用的重要性降低。此外，折扣因子保证了定义 $J_\pi(x)$ 的极限存在且有限（假设阶段费用 g 的取值范围是有界的）。这是一种良好的数学性质，使得带折扣的问题在解析和算法上均易于处理。

因此，某一策略的无穷阶段费用是其有限阶段费用随着阶段数趋于无穷的极限（我们暂且假设该极限存在且为有限值，后续小节中我们再研究此问题）。接下来我们将研究如下三类问题：

(a) 随机最短路径问题（*stochastic shortest path problems*）（英文简称为 SSP）。此时，$\alpha = 1$，但问题中有一个特殊的无费用的终止状态；一旦系统抵达终止状态，就会一直停留在那里，且后续费用均为零。在某些问题中，终止状态可能表示一个我们试图以最小成本达到的目标状态，而在另一些问题中，它可能是一个我们试图尽可能长时间避免到达的状态。我们一般都假设问题结构使得在所有策略下终止都是不可避免的。因此，问题中的有效视界是有限长的，但其具体的取值是随机的，且可能受到所用策略的影响。此外还有一类明显更复杂的随机最短路径问题，其中只有一部分策略（包括所有最优策略）能够保证会抵达终止状态。我们将选择性地讨论这一情况。一些常见的随机最短路径问题属于这一类，例如涉及具有环的图的确定性最短路径问题。

(b) 折扣问题（*discounted problems*）。此时，$\alpha < 1$，且问题中无需终止状态。然而，我们可以很容易地将折扣问题转化为随机最短路径问题。通过引入一个虚拟的终止状态，并认为每阶段的系统状态都有 $1 - \alpha$ 的概率转移到该虚拟状态，从而使状态终止不可避免。鉴于此，关于随机最短路径问题的算法和分析经过简单调整就可以适用于折扣问题。动态规划教材 [Ber17] 对这一转换提供了详细说明，并对具有有限状态数的折扣问题和随机最短路径问题进行了通俗易懂的介绍。

(c) 确定性非负费用问题（*deterministic nonnegative cost problems*）。这里，扰动 w_k 只取一个已知的值。换句话说，系统方程和成本表达式中都不存在扰动，它们现在的形式为

$$x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, \ldots, \tag{1.28}$$

以及

$$J_\pi(x_0) = \lim_{N\to\infty} \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k)\big), \tag{1.29}$$

我们进一步假设存在一个无费用且吸收性的终止状态 t，并且满足

$$g(x, u) \geqslant 0, \qquad \text{对所有} x \neq t,\ u \in U(x), \tag{1.30}$$

以及 $g(t, u) = 0$ 对所有 $u \in U(t)$。这种结构表达了以最小费用到达或逼近 t 这个目标，这是一个经典的控制问题。关于该问题无折扣版本的广泛分析，详见作者论文 [**?**]。

具有有限状态数的折扣随机问题 [也称为折扣马尔可夫决策问题（*disounted Markovian Decision Problems*，简写作 *MDP*）] 在动态规划/强化学习文献中非常常见，特别是由于它们在解析和计算上具有良好的性质。此外，普遍存在一种观点，认为带折扣的马尔可夫决策问题可以用作通用模型，也就是说，在实际应用中，任何其他类型的问题（例如，带有终止状态和/或连续状态空间的无折扣问题）都可以轻松地转换为折扣因子足够接近 1 的带折扣马尔可夫决策问题。然而，由于若干原因，这种观点是值得商榷的：

(a) 确定性模型在许多实际应用中（包括离散优化/整数规划问题）既常见又自然，因此将它们转换为马尔可夫决策问题并没有意义。

(b) 通过某种离散化方法将连续状态问题转换为有限状态问题，涉及一些数学上的微妙问题，这可能导致严重的实施或算法上的复杂性。特别地，无论离散化有多精细，将问题转换为带折扣的马尔可夫决策问题可能会严重扭曲最优解的特性。

(c) 对于某些实际的最短路径问题来说，最终必须到达终止状态是至关重要的。然而，在这类问题中引入折扣因子 α 后，问题的性质可能会从根本上发生改变。特别地，适当的 α 值的临界阈值可能非常接近于 1，而且在实际中可能未知。举一个简单的例子，考虑一个包含状态 1、状态 2 和终止状态 t 的最短路径问题。从状态 1 出发，我们可以以费用 0 到达状态 2；从状态 2 出发，可以以一个很小的费用 $\epsilon > 0$ 返回状态 1，也可以以较高的费用 $C > 0$ 到达终止状态 t。在无限时域内的最优策略是：从状态 1 到状态 2，再从状态 2 到状态 t。现在，假设我们通过引入一个折扣因子 $\alpha \in (0, 1)$ 来对问题进行近似。可以证明，当 $\alpha < 1 - \epsilon/C$ 时，最优策略是无限期地沿着循环 $1 \to 2 \to 1 \to 2$ 运动，而永远不达到终止状态 t；而当 $\alpha > 1 - \epsilon/C$ 时，则会得到最短路径 $2 \to 1 \to t$。因此，折扣问题的解随着 t 的变化而不连续：在某个阈值处，解会发生剧烈变化，而这一阈值通常可能未知。

本书将详细讨论的一类重要问题是具有大量状态的有限状态确定性问题。这些问题的有限阶段版本包括那些具有挑战性的离散优化问题，而这些问题的精确求解在实际中几乎是不可能的。需要牢记的一个重要事实是，我们可以将此类问题转换为在时域末尾设有终止状态的无穷阶段随机最短路径问题，从而使本节的概念框架得以应用。利用强化学习方法，特别是策略前展方法，对离散优化问题进行近似求解将在第 3 章中讨论。

1.4.1 无穷阶段问题方法

关于无穷阶段动态规划问题，已有丰富的理论成果。在本节中，我们将给出一个直观性的总结，其中包括对值迭代和策略迭代这两种主要算法的介绍。更严谨的数学论述（包括证明）将在附录 A 中给出，进一步的讨论和参考文献将在第 1.7 节中提供。

贝尔曼方程与值迭代

关于我们的无穷阶段问题存在若干解析和计算方面的问题。其中许多问题都集中在无穷阶段问题的最优费用函数 J^* 与相应的 N 阶问题的最优费用函数之间的关系上。

具体来说，记 $J_N(x)$ 为涉及 N 个阶段、初始状态为 x、每阶段费用为 $g(x,u,w)$ 且终止费用为零的问题的最优费用。该费用是通过对下面的算法迭代 N 次得到的：

$$J_{k+1}(x) = \min_{u\in U(x)} E_w\Big\{g(x,u,w) + \alpha J_k\big(f(x,u,w)\big)\Big\}, \qquad k = 0,1,\ldots, \tag{1.31}$$

起始于 $J_0(x) \equiv 0$。[21]该算法(1.31)被称为*值迭代*（*value iteration*）算法（简称 VI）。由于给定策略的无穷阶段费用按定义为对应 N 阶问题费用当 $N \to \infty$ 时的极限，因此自然会猜测：

(a) 最优无穷阶段费用是相应的 N 阶段最优费用在 $N \to \infty$ 时的极限；即对于所有状态 x，有

$$J^*(x) = \lim_{N\to\infty} J_N(x). \tag{1.32}$$

(b) 对于所有状态 x，应有

$$J^*(x) = \min_{u\in U(x)} E_w\Big\{g(x,u,w) + \alpha J^*\big(f(x,u,w)\big)\Big\}. \tag{1.33}$$

这是利用式(1.32)在值迭代算法(1.31)中取 $N \to \infty$ 的极限得到的。上述方程称为*贝尔曼方程*（*Bellman's equation*），它实际上是一组方程（每个状态 x 一个方程），其解即为所有状态的最优展望费用。

(c) 如果对于每个状态 x，$\mu(x)$ 能使贝尔曼方程(1.33)右侧取得最小值，那么策略 $\{\mu,\mu,\ldots\}$ 应该是最优的。这种类型的策略称为*稳态*（*stationary*）策略，为简便起见记作 μ。

(d) 稳态策略 μ 的费用函数 J_μ 满足

$$J_\mu(x) = E_w\Big\{g\big(x,\mu(x),w\big) + \alpha J_\mu\big(f(x,\mu(x),w)\big)\Big\}. \tag{1.34}$$

我们可以将其看作是针对一个不同问题的贝尔曼方程(1.33)：对于每个 x，控制约束集 $U(x)$ 仅包含一个控制，即 $\mu(x)$。此外，我们期望 J_μ 能通过值迭代算法在极限下获得，即

$$J_\mu(x) = \lim_{N\to\infty} J_{\mu,N}(x), \qquad \text{对所有} x,$$

[21]这只是第 1.3.1 节中有限阶段动态规划算法，只不过我们为了适应无穷阶段的情形，将时间指标反转了。具体来说，考虑一个含有 N 个阶段的问题，令 $V_{N-k}(x)$ 表示从状态 x 出发、剩余 k 阶段的最优展望费用（且终止费用为 0）。对所有 x，应用动态规划就得到

$$V_{N-k}(x) = \min_{u\in U(x)} E_w\Big\{\alpha^{N-k} g(x,u,w) + V_{N-k+1}\big(f(x,u,w)\big)\Big\}, \quad V_N(x) = 0.$$

通过定义 $J_k(x) = V_{N-k}(x)/\alpha^{N-k}$，我们便得到了值迭代算法(1.31)。

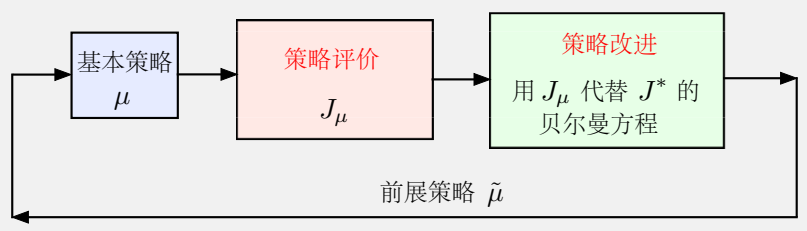


图 1.4.2: 策略迭代作为重复策略前展的示意图。它生成一系列策略，其中序列中的每个策略 μ 作为基本策略，并通过策略前展生成下一个策略 $\tilde{\mu}$。这个前展策略在随后的迭代中又被用作基本策略。

其中 $J_{\mu,N}$ 是通过下式算出的对应于 μ 的 N 阶段费用函数

$$J_{\mu,k+1}(x) = E_w\Big\{g\big(x,\mu(x),w\big) + \alpha J_{\mu,k}\big(f(x,\mu(x),w)\big)\Big\} \tag{1.35}$$

初始条件为 $J_{\mu,0} \equiv 0$ 或某个其他函数；参见式(1.31)-(1.32)。

以上四个结果都可以证明适用于有限状态的折扣问题，并且在合理假设下也适用于有限状态的随机最短路径问题（见附录 A）。只要每阶段费用函数 g 在 (x,u,w) 的取值集合上有界，这些结果也适用于无穷状态的折扣问题，在这种情况下我们还可以证明 J^* 是贝尔曼方程的唯一解。在这些条件下，值迭代算法也是有效的，即使初始函数 J_0 非零，也有 $J_k \to J^*$。选择不同的 J_0 的目的在于能够更快地收敛到 J^*；通常 J_0 越接近 J^*，收敛速度就越快。相关的数学证明可参见多个来源，例如 [Ber12] 的第 1 章或 [Ber19b] 的第 4 章。

对于无折扣问题以及具有无界阶段费用的折扣问题，我们仍可以假设前述四个结果为真。然而，我们也应注意，在不利条件下可能出现异常行为，包括贝尔曼方程解的不唯一性以及值迭代算法在某些初始条件下不收敛到 J^*；详见附录 A 以及著作 [Ber12] 和 [Ber22a]。

需要注意的是，对于无穷阶段问题，还有其他一些重要算法适用于在值空间中进行近似。这些算法包括近似策略迭代、Q-学习、时序差分方法、线性规划及其变种；详见强化学习书籍 [Ber19b]、[Ber20b]。因此，在涉及无穷阶段的情况下，尽管相关数学理论更为复杂，但用于值空间近似的算法选项更加丰富。在本书中，我们仅讨论策略迭代算法的近似形式及其变种，下面将对此进行描述。

策略迭代

适用于无穷阶段问题的一个主要算法是策略迭代（*policy iteration*）（英文简写作 PI）。我们将论证，策略迭代及其变种构成了强化学习中自主学习的基础，即从系统自身运行过程中自主生成的数据中学习，而不是依赖外部提供的数据。图1.4.2将这种方法描述为重复执行策略前展，并指出每次迭代包含两个阶段：

(a) 策略评价（*policy evaluation*），即计算当前（或基本）策略 μ 的费用函数 J_μ。一种可能的方法是求解相应的贝尔曼方程

$$J_\mu(x) = E_w\Big\{g\big(x,\mu(x),w\big) + \alpha J_\mu\big(f(x,\mu(x),w)\big)\Big\}, \qquad \text{对所有} x,$$

参见式(1.34)。然而，对于任一状态 x，$J_\mu(x)$ 的值也可以通过蒙特卡罗仿真来计算，即对从 x 出发的多条随机生成的轨迹的费用进行平均。

(b) 策略改进（*policy improvement*），通过一步前瞻最小化计算“改进”（或前展）策略 $\tilde{\mu}$。对于所有状态 x，

$$\tilde{\mu}(x) \in \arg\min_{u\in U(x)} E_w\Big\{g(x,u,w) + \alpha J_\mu\big(f(x,u,w)\big)\Big\}.$$

我们称 $\tilde{\mu}$ 为“改进策略”，因为通常可以证明

$$J_{\tilde{\mu}}(x) \leqslant J_\mu(x), \qquad \text{对所有} x.$$

这种费用改进性质将在第 A.3 节中证明，并可用于证明在有利条件下（例如，对于有限状态折扣问题；参见动态规划书籍 [Ber12]、[Ber17] 或强化学习书籍 [Ber19b]）策略迭代在有限次迭代后就能产生最优策略。

精确形式的策略前展算法实际上就是策略迭代算法的一次迭代（*a single iteration of the PI algorithm*）。它以给定的基本策略 μ 为起点，产生前展策略 $\tilde{\mu}$。可以将其视为在值空间中进行一步前瞻的近似方法，其中 J_μ 被用作终止费用函数的近似。该方法的优点在于可以在线应用，通过仿真计算所需的 $J_\mu(x)$ 值。相比之下，对于那些具有挑战性的问题（例如涉及神经网络训练的问题），策略迭代的近似形式只能离线实现。

1.4.2 值空间近似——无穷阶段问题

我们在讨论有限阶段问题时使用的值空间近似方法可以自然地扩展应用于无穷阶段问题。在这里，我们用一个近似函数 $\tilde{J}$ 代替 J^*，并在任一状态 x 下，通过一步前瞻最小化生成控制 $\tilde{\mu}(x)$：

$$\tilde{\mu}(x) \in \arg\min_{u\in U(x)} E_w\Big\{g(x,u,w) + \alpha \tilde{J}\big(f(x,u,w)\big)\Big\}. \tag{1.36}$$

这个最小化过程得到一个稳态策略 $\{\tilde{\mu},\tilde{\mu},\dots\}$，其费用函数记为 $J_{\tilde{\mu}}$（即 $J_{\tilde{\mu}}(x)$ 表示从状态 x 开始采用 $\tilde{\mu}$ 所获得的无穷阶段折扣总费用）；参见图 1.4.3。注意，当 $\tilde{J} = J^*$ 时，一步前瞻策略在贝尔曼方程(1.33)中取得最小值，因此该策略将是最优的。这表明应使 $\tilde{J}$ 与 J^* 尽可能接近，而这一点通常是成立的，正如我们后面将要论证的那样。

显然，一个重要的目标是使 $J_{\tilde{\mu}}$ 在某种意义上接近 J^*。然而，对于经典控制问题——这些问题涉及将状态引导并保持在理想的参考状态附近（例如，具有无费用且吸收性的终止状态、而其他状态均有正的费用的问题）——$\tilde{\mu}$ 的稳定性可能是主要目标（*stability of $\tilde{\mu}$ may be a principal objective*）。在本书中，我们将主要讨论这一类问题的稳定性问题，并认为当 $J_{\tilde{\mu}}$ 为实值时，策略 $\tilde{\mu}$ 是稳定的（*we will consider the policy $\tilde{\mu}$ to be stable if $J_{\tilde{\mu}}$ is real-valued*），即对于所有状态 x 都满足

$$J_{\tilde{\mu}}(x) < \infty$$

时，我们称策略 $\tilde{\mu}$ 是稳定的。在某些应用场景中（例如模型预测控制和自适应控制），选择 $\tilde{J}$ 以确保 $\tilde{\mu}$ 的稳定性是一个十分重要的问题，我们将在下一节中，在有限阶段的线性二次型问题中对此进行讨论。

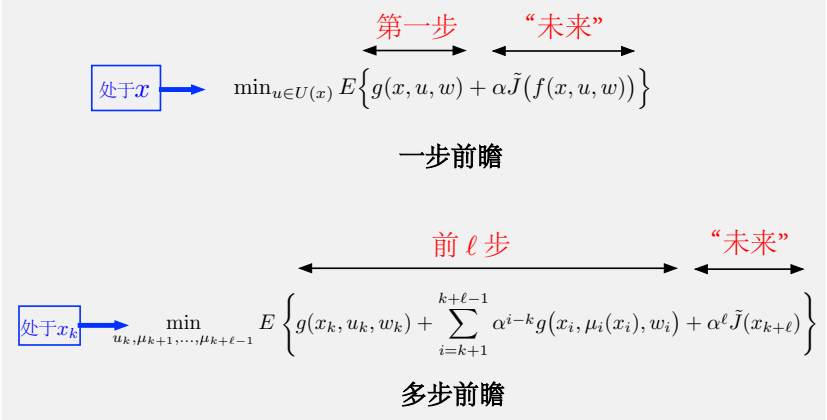


图 1.4.3: 无穷阶段问题中，值空间近似方法的一步前瞻和 ℓ-步前瞻最小化的示意图。在前一种情况下，最小化在状态 x 下产生了一个控制 $\tilde{u}$，从而通过

$$\tilde{\mu}(x)=\tilde{u}$$

定义了一步前瞻策略 $\tilde{\mu}$。在后一种情况下，最小化得到了一个控制 $\tilde{u}_k$ 以及一系列策略 $\tilde{\mu}_{k+1},\ldots,\tilde{\mu}_{k+\ell-1}$。在状态 x_k 处应用控制 $\tilde{u}_k$，而剩余的策略序列 $\tilde{\mu}_{k+1},\ldots,\tilde{\mu}_{k+\ell-1}$ 则被舍弃。该控制 $\tilde{u}_k$ 定义了 ℓ-步前瞻策略 $\tilde{\mu}$。

ℓ-步前瞻

一步前瞻最小化的一个重要扩展是 ℓ-步前瞻。在状态 x_k 下，我们最小化前 ℓ ($\ell>1$) 个阶段的费用，并用函数 $\tilde{J}$ 来近似未来费用（参见图1.4.3下半部分）。[22]这种最小化得到一个控制 $\tilde{u}_k$ 和一系列策略 $\tilde{\mu}_{k+1},\ldots,\tilde{\mu}_{k+\ell-1}$。在状态 x_k 处应用控制 $\tilde{u}_k$，并通过设定 $\tilde{\mu}(x_k)=\tilde{u}_k$，从而定义了 ℓ-步前瞻策略 $\tilde{\mu}(x_k)$，而策略序列 $\tilde{\mu}_{k+1},\ldots,\tilde{\mu}_{k+\ell-1}$ 则被舍弃。实际上，我们可以将 ℓ-步前瞻最小化视为一步前瞻最小化的特例，其中前瞻函数为一个 $(\ell-1)$-阶段动态规划问题的最优费用函数，该问题在经过 $(\ell-1)$ 阶段后，对状态 $x_{k+\ell}$ 采用终止费用 $\tilde{J}(x_{k+\ell})$。

多步前瞻最小化的动机在于，通过增大 ℓ 的值，我们可能不需要过于精确的近似 $\tilde{J}$ 也能取得良好性能。换句话说，在费用函数近似质量相同的情况下，ℓ 越大，性能可能越好。稍后将在第 1.5 节中利用牛顿法的理论直观地解释这一点。特别地，对于阿尔法零国际象棋来说，较长的多步前瞻对于良好的在线表现至关重要。多步前瞻的另一个动机是增强生成的在线策略的稳定性（*enhance the stability properties of the generated on-line policy*），我们将在第 1.5 节中进一步讨论这一点。另一方面，相比于求解式(1.36)的一步前瞻最小化，求解多步前瞻最小化问题耗时更长。

三种近似：最优费用、期望值与前瞻最小化

对于无穷阶段问题，有三个潜在的近似领域：最优费用近似、期望值近似和最小化近似；参见图1.4.4。它们与我们在第 1.3.2 节中讨论的有限阶段问题中相应的近似方法类似。具体来说，我们可能有以下几种近似：

[22]在强化学习文献中，采用多步前瞻最小化（可能还包括截短的策略前展）的在线运行有多种不同的称呼，例如在线搜索、预测学习、从预测中学习（*on-line search, predictive learning, learning from prediction*）等；而在模型预测控制文献中，将前瞻最小化与截短的策略前展结合起来的区间称为预测区间（*prediction interval*）。

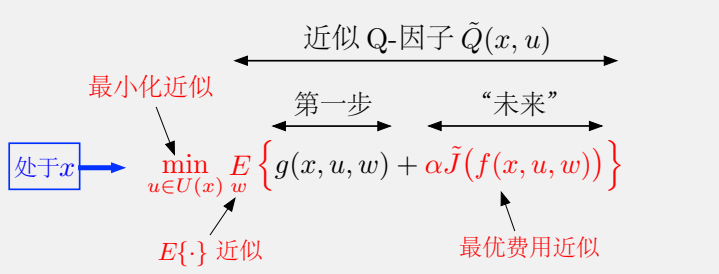


图 1.4.4: 针对无穷阶段问题采用一步前瞻的值空间近似。其中有三个潜在的近似领域，且相关的近似方法可以相互独立地涉及：最优费用近似、期望值近似和最小化近似。

(a) 对最优费用函数 J^* 的终止费用近似 $\tilde{J}$ (*terminal cost approximation $\tilde{J}$ of the optimal cost function J^**)：无穷阶段问题的一个主要优势在于，只需要一个近似费用函数 $\tilde{J}$，而不像 N 阶段问题那样需要 N 个函数 $\tilde{J}_1,\ldots,\tilde{J}_N$。

(b) 对期望运算的近似 (*approximation of the expected value operation*)：该运算可能非常耗时，可以通过各种方式简化。例如，在 ℓ-步前瞻最小化中出现的一些随机量 $w_k, w_{k+1},\ldots,w_{k+\ell-1}$ 可以被替换为确定性量；这又是确定性等效方法的一个例子，我们在第 1.3.2 节中对此进行了讨论。

(c) 对最小化运算的简化 (*simplification of the minimization operation*)：例如，在多智能体问题中，控制由多个组成部分构成，

$$u=(u^1,\ldots,u^m),$$

其中每个分量 u^i 由不同的智能体或决策者选择。在这种情况下，控制空间的规模可能非常庞大，但可以通过后面讨论的方法进行简化（例如，依次选择各个分量，即一次由一个智能体选择）。这将构成我们处理多智能体问题方法的核心；参见第 2.9 节和第 4.3 节。

接下来，我们将简要介绍选择终止费用函数近似的各种方法。

为在线执行构造终止费用近似

值空间近似中的一个主要问题是构造一个合适的费用函数近似 $\tilde{J}$。这可以通过多种不同的方法来实现，由此得到一些主要的强化学习方法。

例如，$\tilde{J}$ 可以通过复杂的离线训练方法构造。另外，也可以按需利用截短策略前展方法在线获得近似值 $\tilde{J}(x)$：从状态 x 出发运行一个离线得到的策略足够多的步数，并辅以合适的、也许相对简单的终止费用近似。

为了便于理解，我们简要描述四种广泛的近似方法。稍后我们将再次回顾这些方法，同时我们也推荐参考强化学习和近似动态规划的文献，以获得更详细的讨论。

(a) 离线问题近似 (*off-line problem approximation*)：在这种方法中，函数 $\tilde{J}$ 是离线计算得到的，该函数是一个简化的优化问题的最优或近似最优费用函数。这个简化问题在计算上更为方便。简化措施可能包括利用可分解结构、缩小状态空间规模、忽略部分

约束以及排除各种不确定性。例如，我们可以考虑将一个相关的确定性问题的成本函数作为 $\tilde{J}$，该问题通过某种形式的确定性等价近似获得，从而允许利用基于梯度的最优控制方法或最短路径类型的方法来计算 $\tilde{J}$。

一种主要的问题近似方法是聚集（*aggregation*），这在本书的第 4.6 节、书籍 [Ber12]、[Ber19b] 以及论文 [Ber18]、[Ber19a] 中都有描述。聚集提供了一种系统性的方法，通过将状态分组为相对较少的子集（称为聚集状态），来简化给定的问题。我们可以利用精确的动态规划方法（可能包括仿真）来计算这个更为简单的聚集问题的最优费用函数。然后，基于该费用函数，我们可以通过某种插值方法获得原问题的最优成本函数 J^* 的近似 $\tilde{J}$。

(b) *在线仿真*（*on-line simulation*）：该近似方法适用于针对随机问题的策略前展算法中。此时，我们利用蒙特卡罗仿真以及某个次优策略 μ（基本策略）来计算（所需的）$\tilde{J}(x)$，从而使得 $\tilde{J}(x)$ 完全或近似等于 $J_\mu(x)$ 的值。策略 μ 可以通过任何方法获得，例如基于启发式推理的方法（如旅行商问题示例 1.2.3 中的情况），或者基于更有原则的方法进行离线训练，如近似策略迭代或策略空间近似。请注意，虽然仿真耗时，但它非常适合并行计算。此外，通过使用确定性等效近似方法，还可以简化仿真过程。

(c) *在线近似优化*（*on-line approximate optimization*）：这种方法涉及求解一个适当构造的较短时域问题，并采用简单的终止费用近似。此类方法既可以看作是值空间中多步前瞻的近似方法，也可以视为一种策略前展算法。该方法常用于模型预测控制（英文简写作 MPC）。

(d) *参数化费用近似*（*parametric cost approximation*），此时 $\tilde{J}$ 是从给定的参数化函数族 $J(x,r)$ 中获得的，其中 r 是一个由适当算法选出的参数向量。该参数化函数族通常涉及利用 x 的显著特性，即所谓*特征*（*features*），这些特征可以通过对当前问题的深入理解获得，或者通过使用训练数据和某种形式的神经网络来提取（参见第 3 章）。

这种方法包括近似形式的策略迭代，正如在第 1.1 节中讨论与国际象棋和双陆游戏相关内容时所提到的。策略迭代算法中的策略评价部分可以通过利用诸如神经网络等近似架构来近似当前策略的费用函数（参见第 5 章）来实现。也可以采用随机迭代算法，例如 TD(λ)、LSTD(λ) 和 LSPE(λ)，这些方法在动态规划书籍 [Ber12] 和强化学习书籍 [Ber19b] 中均有描述。这些方法在本课程中属于次要内容，因此不会进行详细讨论。不过，我们需要注意的是，近似策略迭代方法不仅能够得到一个参数化的近似成本函数 $J(x,r)$，还能得到一个次优策略，而该策略可以通过使用（可能经过截短的）策略前展在线改进。

除了近似策略迭代方法之外，参数化近似费用函数 $J(x,r)$ 还可以通过 Q-学习、线性规划和聚集方法等离线方法获得，这些方法也在书籍 [Ber12] 和 [Ber19b] 中有所讨论。

我们还应提到，对于具有特殊结构的问题，可以选择合适的 $\tilde{J}$ 来简化一步前瞻最小化 [式(1.36)]。事实上，在有利条件下，前瞻最小化可以用解析表达来求解。举例来说，当系统是非线性的，但控制在系统方程中以线性形式出现，而在费用函数中呈二次形式，同时终止费用近似也是二次的，那么一步前瞻最小化就可以解析地求解，因为它涉及的函数是关于 u 的二次函数。

从离线训练到在线执行

一般来说，离线训练会产生仅仅一个费用近似（例如 TD-Gammon 的情况），或仅仅一个策略（例如通过策略空间近似/策略梯度方法），或者两者兼有（如阿尔法零的情况）。我们在本节中已经讨论了基于 $\tilde{J}$ 的一步前瞻和多步前瞻的值空间近似方案；参见图1.4.3。现在让我们考虑一些额外的可能性，这些方法涉及使用离线获得的策略 μ（可能还包括终止费用近似）。以下是一些主要的可能性：

(a) 给定一个离线获得的策略 μ（*given a policy μ that has been obtained off-line*），我们可以将该策略的成本函数 J_μ 用作终端成本近似 $\tilde{J}$。对于一步前瞻情况，这需要进行策略评价操作，并可以在线实现，即只计算（可能通过仿真）所需的

$$E\Big\{J_\mu\big(f(x_k,u_k,w_k)\big)\Big\}$$

的值 [参见式(1.36)]。而对于 ℓ-步前瞻情况，则需要计算所有从状态 x_k 出发在 ℓ 步内可达的状态 $x_{k+\ell}$ 的

$$E\big\{J_\mu(x_{k+\ell})\big\}$$

的值。这是策略前展的最简单形式，并且只需要离线构造策略 μ。

(b) 给定一个离线获得的终止费用近似 $\tilde{J}$（*given a terminal cost approximation $\tilde{J}$ that has been obtained off-line*），我们可以在需要时利用它在线地快速计算相应的一步或多步前瞻策略 $\tilde{\mu}$ 所对应的控制。随后，该策略 $\tilde{\mu}$ 又可如上文 (a) 部分所示用于策略前展。在这种方案的截短变体中，我们还可以用 $\tilde{J}$ 来近似策略前展截去的尾部费用（例如，基于策略前展的 TD-Gammon 算法就是一个例子）。

(c) 给定一个策略 μ 和一个终端成本近似 $\tilde{J}$（*given a policy μ and a terminal cost approximation $\tilde{J}$*），我们可以将它们结合使用于截短策略前展方案中，其中前展过程采用策略 μ 进行仿真，而截去的尾部的费用则通过函数 $\tilde{J}$ 进行近似。这与上面 (b) 中提到的截短策略前展方案类似，不同之处在于这里策略 μ 是离线计算得到的，而不是在线利用 $\tilde{J}$ 和一步或多步前瞻计算得到的。

上述三种可能性是将离线训练结果用于在线运行方案的主要方式。当然，也存在其他变体，通过离线计算额外信息来辅助和/或加速在线运行算法。例如，在模型预测控制中，除了终止费用近似之外，还可能需要离线计算一个目标管道，以确保某些状态约束能在在线运行时得到满足；有关模型预测控制的讨论，请参见第 2.11 节。在特定应用的背景下，还会提到此类其他例子。

最后，需要注意的是，尽管我们主要介绍了在值空间中使用费用函数近似的方法，但我们的讨论同样适用于 Q-因子近似，其涉及的函数为

$$\tilde{Q}(x,u)\approx E\Big\{g(x,u,w)+\alpha J^*\big(f(x,u,w)\big)\Big\}.$$

相应的一步前瞻方案形式为

$$\tilde{\mu}(x)\in\arg\min_{u\in U(x)}E\Big\{g(x,u,w)+\alpha\min_{u'\in U(f(x,u,w))}\tilde{Q}\big(f(x,u,w),u'\big)\Big\};\qquad(1.37)$$

参见式(1.36)。上述方程右侧的第二项表示费用函数近似

$$\tilde{J}\big(f(x,u,w)\big) = \min_{u' \in U(f(x,u,w))} \tilde{Q}\big(f(x,u,w),u'\big).$$

在“无模型”情况下，Q-因子的使用非常常见，此时利用计算机仿真器生成扰动 w 的样本以及对应的 g 和 f 的值。然后，在通过离线训练获得 $\tilde{Q}$ 后，必须借助仿真器在线上执行式(1.37)中的一步前瞻最小化。

1.4.3 理解值空间近似

现在我们将讨论一些目标，以期深入理解值空间近似的过程。显然，用一个尽可能接近 J^* 的函数 $\tilde{J}$ 来近似 J^* 是合理的。然而，我们也应尝试定量地理解 $\tilde{J}$ 与 $J_{\tilde{\mu}}$ 之间的关系，后者是基于 $\tilde{J}$ 通过一步前瞻（或多步前瞻）所得策略 $\tilde{\mu}$ 的费用函数。在这方面，以下几个有趣的问题值得探讨：

(a) 离线训练的质量如何影响前瞻策略 $\tilde{\mu}$ 的质量？(*How is the quality of the lookahead policy $\tilde{\mu}$ affected by the quality of the off-line training?*) 一个相关问题是，在给定的近似架构下，我们应当花费多大的精力通过更长、更精细的训练过程来改进 $\tilde{J}$? 在这方面，一个提供诸多启发的基本事实是，$J_{\tilde{\mu}}$ 是从 $\tilde{J}$ 出发，针对贝尔曼方程(1.33)采用牛顿法的一步迭代[23]所得到的结果（$J_{\tilde{\mu}}$ *is the result of a step of Newton' s method that starts at $\tilde{J}$ and is applied to the Bellman Eq.* (1.33)）。这将成为我们下一节讨论的重点，也是笔者在著作 [Ber20b] 和 [Ber22b] 中阐述的一个重要观点。

一个相关的事实是，在采用多步前瞻进行值空间近似时，$J_{\tilde{\mu}}$ 是从对 $\tilde{J}$ 进行多次值迭代所得到的函数开始，通过牛顿法一步迭代得到的结果（$J_{\tilde{\mu}}$ *is the result of a step of Newton' s method that starts at the function obtained by applying multiple value iterations to $\tilde{J}$*）。

(b) 多步前瞻实现中的简化会如何影响 $J_{\tilde{\mu}}$？(*How do simplifications in the multistep lookahead implementation affect $J_{\tilde{\mu}}$?*) 值空间近似的牛顿法步骤解读为我们提供了一个重要的见解，即多步前瞻的初始步骤具有特殊的性质。特别地，只有第一步起到了牛顿法的一步迭代的作用，且需要精确实现。随后的步骤是值迭代，仅用于提高牛顿法迭代起始点的质量，因此它们的精确实现并不是至关重要的（*their precise implementation is not critical*）。

这一思路表明，在多步前瞻策略的实现中，对第一步之后的前瞻步骤进行简化，其性能损失相对较小（如果有的话甚至可以忽略不计）。这种简化的重要例子包括采用确定性等价方法（参见第 4.1.2 节）以及对前瞻树进行修剪（参见第 3.4 节）。从实际角度看，在多步前瞻的第一步之后进行简化可以节省大量在线计算量，而这些计算资源可以有效地用于延长前瞻的步数。

(c) 策略 $\tilde{\mu}$ 何时稳定？(*When is $\tilde{\mu}$ stable?*) 在许多控制应用中，稳定性问题非常重要，因为目标是将状态保持在某个参考点或轨迹附近。实际上，在这类应用中，稳定性是主

[23] 在本书中，“one step of Newton's method” 译作 “牛顿法的一步迭代”，其简写 “Newton step” 则译作 “牛顿迭代”。——译者注

要关注的问题，而最优性则相对次要。在此类情境下，我们关注的是刻画那些能够使 $\tilde{\mu}$ 稳定的终端成本近似 $\tilde{J}$ 的集合。

(d) 前瞻最小化或截短策略前展的步数如何影响多步前瞻策略 $\tilde{\mu}$ 的稳定性和性能？(*How does the length of lookahead minimization or the length of the truncated rollout affect the stability and quality of the multistep lookahead policy $\tilde{\mu}$?*) 通常认为延长前瞻步长有利于提高多步前瞻策略的性能质量，但实际上它也对多步前瞻策略的稳定性也有积极的影响，我们对此机制非常感兴趣。

在接下来的讨论中，我们将牢记这些问题。特别是在下一节中，我们将在简单且方便的线性二次型问题的背景下讨论这些问题。然而，借助抽象的动态规划理论，这些结论可以推广应用于更一般的情形；参见附录 B。关于这些问题更详细和广泛的讨论与分析，请参阅作者的著作 [Ber20b] 和 [Ber22b]。

1.5 牛顿法——线性二次型问题

接下来，我们将尝试在一个重要的确定性问题背景下理解贝尔曼方程的特性、值空间中的近似，以及值迭代和策略迭代算法的表现。该问题属于经典的连续状态问题，其中系统是线性的，没有控制约束，并且费用函数是非负的二次函数。虽然这个问题可以解析求解，但它为直观理解贝尔曼方程及其算法求解（无论是精确求解还是近似求解）提供了一个独特而富有启发性的情境。

在一般形式下，该问题涉及的系统为

$$x_{k+1} = Ax_k + Bu_k,$$

其中 x_k 和 u_k 分别属于欧几里得空间 $\Re^n$ 和 $\Re^m$，A 是 $n \times n$ 矩阵，而 B 是 $n \times m$ 矩阵。假设不存在控制约束。每阶段的费用为二次型，形式为

$$g(x, u) = x'Qx + u'Ru,$$

其中 Q 和 R 分别说维度为 $n \times n$ 和 $m \times m$ 的正定对称矩阵（本文中所有有限维向量均视为列向量，角分符号表示转置）。这一问题的分析已经非常成熟，并在多本控制理论教材中给出了证明，包括笔者的动态规划书籍 [Ber17] 和 [Ber12]。

为了简化起见，接下来我们将仅关注该问题的一维版本，其中系统的形式为

$$x_{k+1} = ax_k + bu_k; \tag{1.38}$$

参见示例 1.3.1。在这里，状态 x_k 和控制 u_k 均为标量，系数 a 和 b 也都是标量，并且 $b \neq 0$。费用函数是无折扣的，其形式为

$$\sum_{k=0}^{\infty} (qx_k^2 + ru_k^2), \tag{1.39}$$

其中 q 和 r 是正的标量。一维情况为我们提供了一个便捷且富有启发性的分析方法，用以探讨对我们来说至关重要的算法问题。该分析可以推广到多维线性二次型问题乃至更一般的情形，但那将需要更为严谨的数学分析。

黎卡提方程及其理论依据

通过将有限时域的例子1.3.1中的结果取极限（当时域长度趋于无穷大时），可以得到本问题的解析解。特别地，我们可以证明最优代价函数的期望形式为如下二次型：

$$J^*(x) = K^* x^2, \tag{1.40}$$

其中标量 K^* 满足如下方程：

$$K = F(K), \tag{1.41}$$

而函数 F 定义为：

$$F(K) = \frac{a^2 rK}{r + b^2 K} + q. \tag{1.42}$$

此方程即为式(1.19)的极限形式。

此外，最优策略是如下线性形式：

$$\mu^*(x) = L^* x, \tag{1.43}$$

其中标量 L^* 由以下公式给出：

$$L^* = -\frac{abK^*}{r + b^2 K^*}. \tag{1.44}$$

为了验证方程式(1.41)-(1.44)的合理性，我们将证明由式(1.40)定义的函数 J^* 满足如下贝尔曼方程：

$$J(x) = \min_{u \in \Re} \left\{ qx^2 + ru^2 + J(ax + bu) \right\} \tag{1.45}$$

并且由式(1.43)-(1.44)定义的策略 $\mu^*(x)$ 能够在 $J = J^*$ 时对每个 x 取得上式的最小值。事实上，对于任意形如 $J(x) = Kx^2$ 且 $K \geqslant 0$ 的二次型费用函数，贝尔曼方程(1.45)中的最小化问题可以写成：

$$\min_{u \in \Re} \left\{ qx^2 + ru^2 + K(ax + bu)^2 \right\}. \tag{1.46}$$

因此，该问题涉及对关于 u 的一个正定二次函数的最小化，可以得到解析解。通过将式(1.46)括号内的表达式对 u 求导并令其为零，可得：

$$0 = 2ru + 2bK(ax + bu),$$

由此求出最小化的控制变量及相应的策略为：

$$\mu_K(x) = L_K x, \tag{1.47}$$

其中

$$L_K = -\frac{abK}{r + b^2 K}. \tag{1.48}$$

通过将上述控制代入，式(1.46)中的最小化表达式变为

$$\left(q + rL_K^2 + K(a + bL_K)^2 \right) x^2.$$

经过直接的代数运算，并利用式(1.48)中 L_K 的定义，可以验证此表达式能够表示为 $F(K)x^2$ 的形式，其中函数 F 即为式(1.42)所定义的函数。因此，当 $J(x) = Kx^2$ 时，贝尔曼方程(1.45)可表示为

$$Kx^2 = F(K)x^2,$$

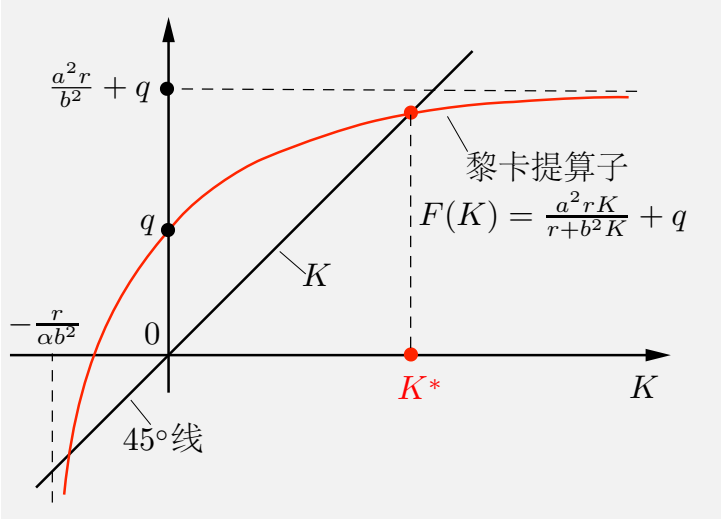


图 1.5.1: 线性二次型问题中黎卡提方程(1.41)-(1.42)解的构造方法图示。最优费用函数为 $J^*=K^*x^2$, 其中标量 K^* 满足不动点方程 $K=F(K)$, 而函数 F 是由以下式子定义的黎卡提算子:

$$F(K)=\frac{a^2rK}{r+b^2K}+q.$$

注意, 函数 F 在区间 $(-r/b^2,\infty)$ 上为凹函数并单调递增, 且当 $K\to\infty$ 时趋于 "平缓", 如图所示。此外, 二次型黎卡提方程 $K=F(K)$ 还有另一个解, 记为 $\bar{K}$, 但该解为负值, 因此在实际问题中并无意义。

等价地, 即为 $K=F(K)$ [参见式(1.41)]。

综上所述, 当我们将考虑范围限定在形如 $J(x)=Kx^2$ (其中 $K\geqslant 0$) 的二次型函数时, 贝尔曼方程(1.45)便等价于下列方程:

$$K=F(K)=\frac{a^2rK}{r+b^2K}+q. \tag{1.49}$$

我们将此方程称为黎卡提方程 (*Riccati equation*), [24]并将函数 F 称作黎卡提算子 (*Riccati operator*)。[25]此外, 对应于解 K^* 的策略由式(1.47)-(1.48)给出, 并且能够使得贝尔曼方程中的表达式达到最小值, 即如式(1.43)-(1.44)所示。

如图1.5.1所示, 黎卡提方程可以通过图解的方式直观展示并求解。由图可见, 对应于最优费用函数 J^* [参见式 (1.40)] 的二次型系数 K^*, 是在非负实数轴上黎卡提方程 $K=F(K)$ 的唯一解。

---

[24]这是黎卡提微分方程的一种代数形式, 最初的一维形式由雅各布 · 黎卡提伯爵 (count Jacopo Riccati) 在 18 世纪 (1700 年代) 提出, 并在控制理论中发挥了重要作用。该方程的微分和差分矩阵形式已经得到了广泛而深入的研究; 相关内容可参阅 Lancaster 与 Rodman 所著的书籍 [LR95], 以及由 Bittanti、Laub 与 Willems 编辑的论文集 [BLW91], 后者中还包含了一篇由 Bittanti 撰写的历史回顾 [Bit91], 讲述了黎卡提非凡的人生经历和学术成就。

[25]黎卡提算子是贝尔曼算子 (记为 T) 的一种特殊情形。贝尔曼算子将一个函数 J 映射为贝尔曼方程右侧的形式:

$$(TJ)(x)=\min_{u\in U(x)}E_w\Big\{g(x,u,w)+\alpha J\big(f(x,u,w)\big)\Big\},\qquad \text{对所有}x.$$

因此, 贝尔曼算子 T 把一个关于 x 的函数 J 转换成另一个同样关于 x 的函数 TJ。通过贝尔曼算子, 可以对问题的数据进行简洁、抽象的描述, 这也是抽象动态规划理论的基础 (参见作者的专著 [Ber22a] 和 [Ber22b])。我们可以将黎卡提算子视为贝尔曼算子在关于 x 的二次型函数子空间中的一种限制形式。

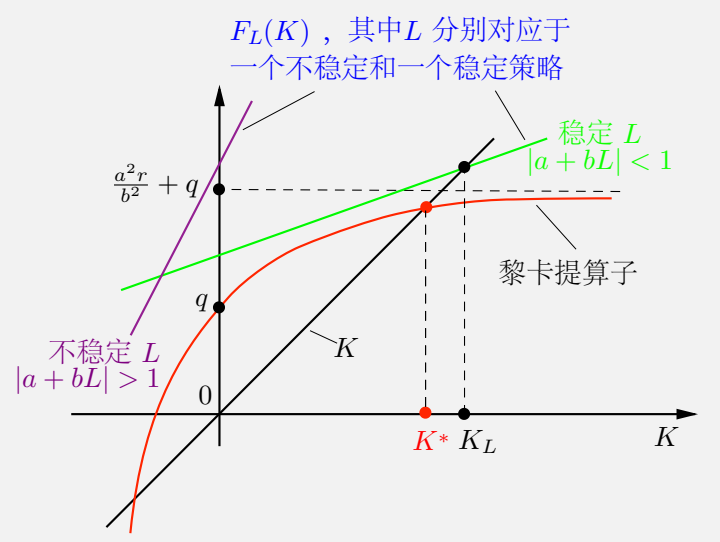


图 1.5.2: 线性策略 $\mu(x)=Lx$ 对应费用函数的构造方法图示，其中该策略为稳定的，即满足 $|a+bL|<1$。该策略对应的费用函数 $J_\mu(x)$ 具有如下形式

$$J_\mu(x)=K_Lx^2,$$

其中 K_L 为线性方程 $K=F_L(K)$ 的唯一解。此处，

$$F_L(K)=(a+bL)^2K+q+rL^2,$$

即为对应于策略 $\mu(x)=Lx$ 的黎卡提方程算子。若策略 μ 不稳定，即 $|a+bL|\geqslant 1$，则对所有非零 x 有 $J_\mu(x)=\infty$。虽然此时方程 $K=F_L(K)$ 仍有解，但在本问题的情境下已无实际意义。

稳定线性策略下的黎卡提方程

我们还可以描述一个线性策略 $\mu(x)=Lx$ 的费用函数，其中策略是稳定的，即标量 L 满足 $|a+bL|<1$ 使得对应的闭环系统

$$x_{k+1}=(a+bL)x_k$$

是稳定的（状态 x_k 在 $k\to\infty$ 时收敛到 0）。具体来说，我们可以证明该策略的费用函数具有如下形式

$$J_\mu(x)=K_Lx^2,$$

其中系数 K_L 满足方程

$$K=F_L(K), \tag{1.50}$$

而函数 F_L 的定义为

$$F_L(K)=(a+bL)^2K+q+rL^2. \tag{1.51}$$

该方程称为稳定策略 $\mu(x)=Lx$ 的黎卡提方程（*Riccati equation for the stable policy* $\mu(x)=Lx$），如图1.5.2所示。注意到 F_L 是线性的，且其线性系数 $(a+bL)^2$ 严格小于 1。因此，表示函数 F_L 图像的直线与 45 度线必然存在唯一交点，从而确定了二次型费用函数的系数 K_L。

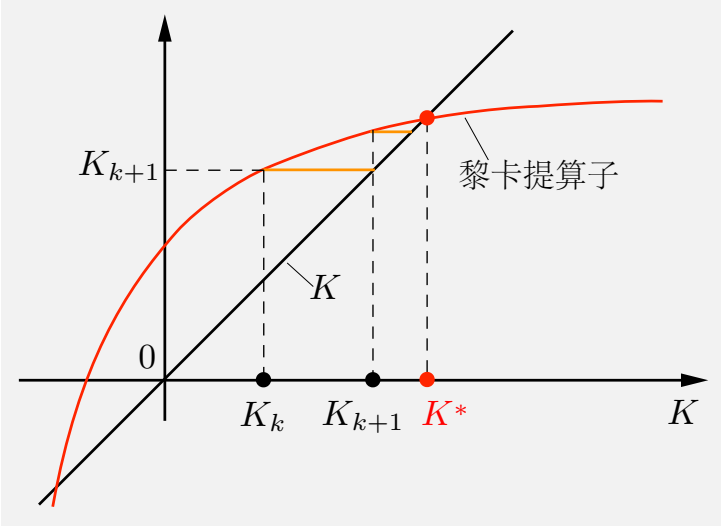


图 1.5.3: 线性二次型问题的值迭代图示。其形式为 $K_{k+1} = F(K_k)$，而 F 是黎卡提算子，其表达式为

$$F(K) = \frac{a^2 rK}{r + b^2 K} + q.$$

该算法从任意 $K_0 \geqslant 0$ 出发均收敛到 K^*。

针对策略 $\mu(x) = Lx$ 的黎卡提方程(1.50)-(1.51)可以通过验证其实际上是该策略的贝尔曼方程来说明合理性，即

$$J(x) = (q + rL^2)x^2 + J\big((a + bL)x\big),$$

[参见式(1.34)]。在我们将函数限定为形如 $J(x) = Kx^2$ 的二次函数时，即得此黎卡提方程。

但需要注意的是，只有当策略 $\mu(x) = Lx$ 是稳定的情况下，$J_\mu(x) = K_L x^2$ 才是黎卡提方程(1.50)-(1.51)的解。如果策略 μ 不稳定，即满足 $|a + bL| \geqslant 1$，由于 $q > 0$ 和 $r > 0$，此时对于所有非零的 x，都有 $J_\mu(x) = \infty$。虽然此时黎卡提方程(1.50)-(1.51)依然存在解，但此解为负值，在我们的情境下没有实际意义。

值迭代

对于我们的线性二次型问题，其值迭代算法定义为

$$J_{k+1}(x) = \min_{u \in \Re} \big\{qx^2 + ru^2 + J_k(ax + bu)\big\}.$$

当 J_k 为二次型，即 $J_k(x) = K_k x^2$（其中 $K \geqslant 0$）时，可以看出一步值迭代计算出 J_{k+1} 的也为二次型，其形式为 $J_{k+1}(x) = K_{k+1}x^2$, 其中

$$K_{k+1} = F(K_k),$$

而 F 即为式(1.49)中定义的黎卡提算子。该算法如图1.5.3所示。从图中可以看出，从任意 $K_0 \geqslant 0$ 出发，算法生成的非负标量序列 $\{K_k\}$ 收敛于 K^*。

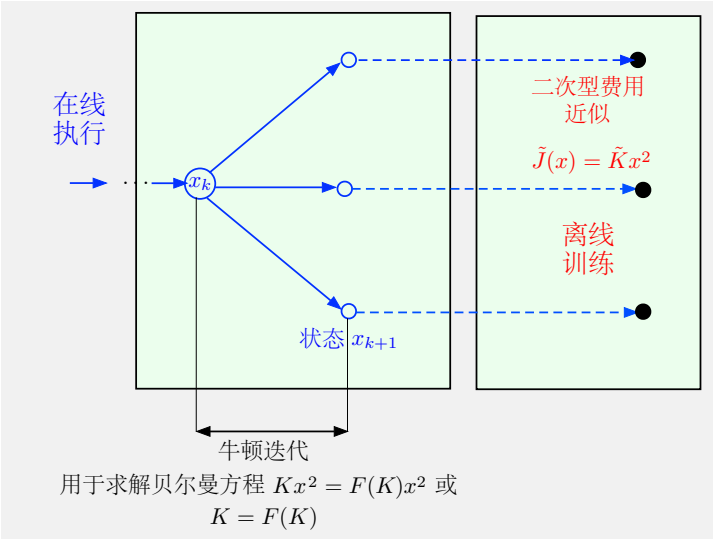


图 1.5.4: 采用一步前瞻的值空间近似方法的的图示，作为牛顿法的一步迭代，该方法将终止费用函数函数近似 $\tilde{J}$ 映射为一步前瞻策略的费用函数 $J_{\tilde{\mu}}$。

1.5.1 值空间近似图示——稳定区域

利用黎卡提方程可以对值空间中的近似进行有助于理解的可视化展示。尽管这种可视化方法专门针对线性二次型问题，但它与更一般无穷阶段问题的相关可视化方法是一致的；这一主题在接下来的内容中将反复出现。特别地，在书籍 [Ber20b] 和 [Ber22b] 中，定义贝尔曼方程的贝尔曼算子被用来取代定义黎卡提方程的黎卡提算子。

总之，我们的目标是说明：

(a) 在采用一步前瞻的值空间近似可以看作是求解贝尔曼方程的牛顿法的一步迭代，并且它将终止费用函数近似 $\tilde{J}$ 映射为一步前瞻策略的费用函数 $J_{\tilde{\mu}}$；参见图1.5.4。

(b) 利用多步前瞻和截短策略前展的值空间中近似方法可以看作求解贝尔曼方程的牛顿迭代，并将以终止费用函数近似 $\tilde{J}$ 为起点的多次值迭代的结果映射为多步前瞻策略的费用函数 $J_{\tilde{\mu}}$；参见图1.5.5。

我们的推导将以一维线性二次型问题为例进行，但实际上适用范围远不止于此（*applies far more generally*）。原因在于，贝尔曼方程在动态规划中具有普遍适用性，并且对应的贝尔曼算子具有凹性特性，非常适合应用牛顿法；详情参见书籍 [Ber20b] 和 [Ber22b]，其中首次详细阐述了值空间近似与牛顿法之间的联系。

让我们考虑采用任意形如 $\tilde{J}(x) = Kx^2$ 的终止费用函数的一步前瞻最小化，其中 $K \geqslant 0$。我们在式(1.47)-(1.48)中，通过最小化当 $J(x) = Kx^2$ 时贝尔曼方程右侧的表达式

$$\min_{u \in \Re} \left\{ qx^2 + ru^2 + K(ax + bu)^2 \right\}$$

推导出了对应的一步前瞻策略 $\mu_K(x)$。我们可以将此最小化过程拆分为两个连续的最小化步骤，如下所示：

$$F(K)x^2 = \min_{L \in \Re} \min_{u = Lx} \left\{ qx^2 + ru^2 + K(ax + bu)^2 \right\} = \min_{L \in \Re} \left\{ q + bL + K(a + bL)^2 \right\} x^2.$$

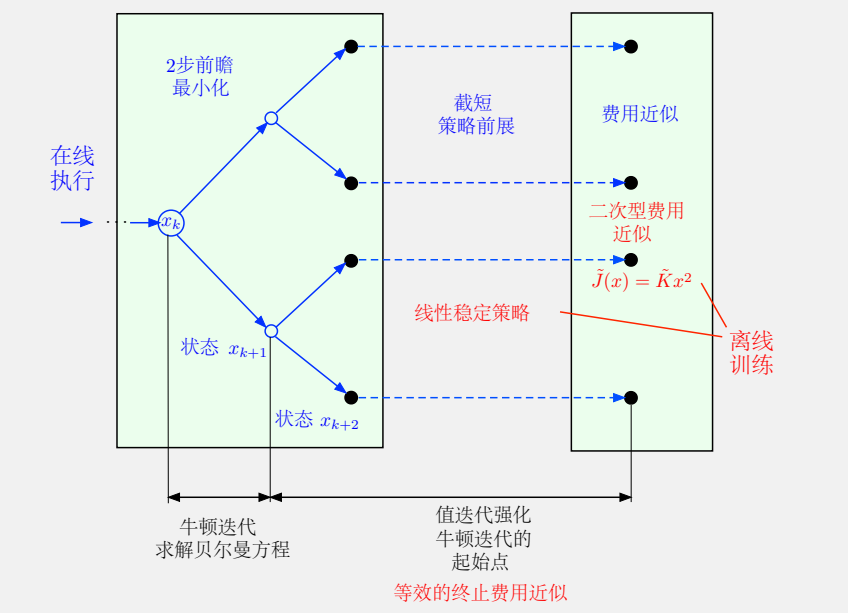


图 1.5.5: 采用多步前瞻和截短策略前展的值空间近似方法作为牛顿法的一步迭代的示意图。该牛顿迭代将以终止费用函数近似 $\tilde{J}$ 为起点的多次值迭代结果映射为多步前瞻策略的费用函数 $J_{\tilde{\mu}}$。

由此可得

$$F(K) = \min_{L \in \Re} F_L(K). \tag{1.52}$$

其中 $F_L(K)$ 的定义为

$$F_L(K) = (a + bL)^2 K + q + bL. \tag{1.53}$$

图1.5.6说明了关系式(1.52)-(1.53)，并展示了如何将黎卡提算子 F 的图像表示为当 L 遍历整个实数时线性算子 F_L 的下包络（*the graph of the Riccati operator F can be obtained as the lower envelope of the linear operators F_L, as L ranges over the real numbers*）。

一步前瞻最小化与牛顿法

现在我们将终端代价函数近似固定为某个 $\tilde{K}x^2$（其中 $\tilde{K} \geqslant 0$），并考虑相应的一步前瞻策略，记作 $\tilde{\mu}$。图1.5.7展示了相应的线性函数 $F_{\tilde{L}}$，并显示其图像在点 $\tilde{K}$ 处是 F 图像的切线 [参见图1.5.6和式(1.53)]。

因此，函数 $F_{\tilde{L}}$ 可视为在点 $\tilde{K}$ 处对 F 的线性化，并定义了一个线性化问题：求解方程

$$K = F_{\tilde{L}}(K) = q + b\tilde{L}^2 + K(a + b\tilde{L})^2.$$

此处的关键在于，该线性方程的解，记作 $K_{\tilde{L}}$，与从点 $\tilde{K}$ 出发，通过牛顿法的一步迭代求解黎卡提方程所得到的解相同（*the solution of this equation, denoted $K_{\tilde{L}}$, is the same as the one obtained from a single iteration of Newton's method for solving the Riccati equation, starting from the point $\tilde{K}$*）。这一点在图1.5.7中有所说明，并且在习题 1.5 中也给出了理论上的证明。

为了说明这种关联，我们注意到牛顿法求解形如 $y = T(y)$ 的不动点问题（其中 y 是一个 n 维向量）的经典形式是如下：在当前迭代点 y_k 处，我们对 T 进行线性化，并求解相

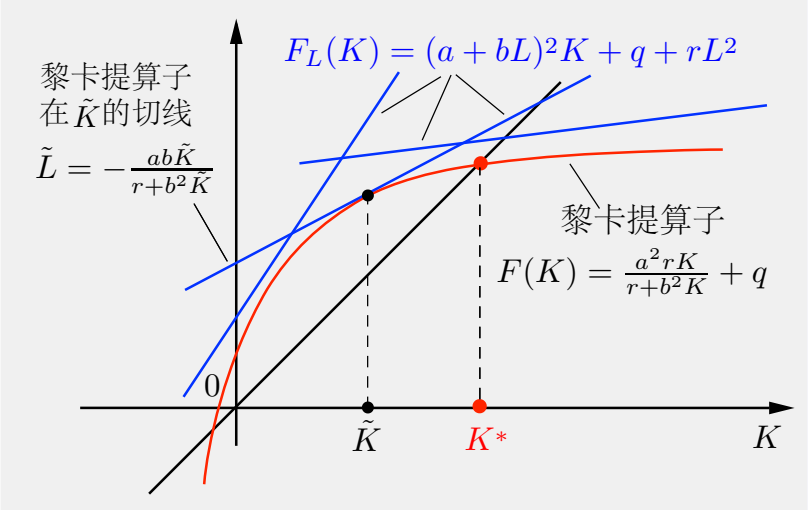


图 1.5.6: 图示了如何将黎卡提算子 F 的图像表示为当 L 遍历实数时一族线性算子

$$F_L(K) = (a+bL)^2K + q + bL$$

的下包络。也就是说，

$$F(K) = \min_{L\in\Re} F_L(K);$$

参见式(1.52)。此外，对于任意固定的 $\tilde{K}$，取得最小值的标量 $\tilde{L}$ 由

$$\tilde{L} = -\frac{ab\tilde{K}}{r+b^2\tilde{K}}$$

给出；参见式(1.48)，并且对应于 $\tilde{L}$ 的 $F_{\tilde{L}}$ 图像在 $\tilde{K}$ 处与 F 的图像相切，如图所示。

应的线性不动点问题以得到 y_{k+1}。假设 T 可微，则利用一阶泰勒展开可得到其线性化表达式：

$$y_{k+1} = T(y_k) + \frac{\partial T(y_k)}{\partial y}(y_{k+1} - y_k),$$

其中 $\partial T(y_k)/\partial y$ 是在向量 y_k 处计算的 $n \times n$ 的雅可比矩阵，如图 1.5.7所示。

牛顿法最常被提及的收敛速率性质是二次收敛。该性质说明在解 y^* 附近，比值

$$\frac{\|y_{k+1} - y^*\|}{\|y_k - y^*\|^2}$$

是有界的，其中 $\|\cdot\|$ 表示欧几里得范数，并且这一性质成立的前提是雅可比矩阵存在且满足利普希茨连续性（参见 [Ber16] 第 1.4 节）。此外，还有一些牛顿法的扩展方法，这些方法在当前迭代点求解线性化系统时放宽了对可微性的要求，可以仅要求分段可微和/或分量凹性，同时仍保持二次或类似快速的超线性收敛性；参见附录 B。

另外需要注意的是，如果一步前瞻策略是稳定的，即满足 $|a+b\tilde{L}| < 1$，那么 $K_{\tilde{L}}$ 就是其费用函数的二次型系数，即

$$J_{\tilde{\mu}}(x) = K_{\tilde{L}}x^2.$$

原因在于 $J_{\tilde{\mu}}$ 满足策略 $\tilde{\mu}$ 的贝尔曼方程。另一方面，如果 $\tilde{\mu}$ 不稳定，则考虑到每阶段正定的二次费用，对于所有 $x \neq 0$ 都有 $J_{\tilde{\mu}}(x) = \infty$。

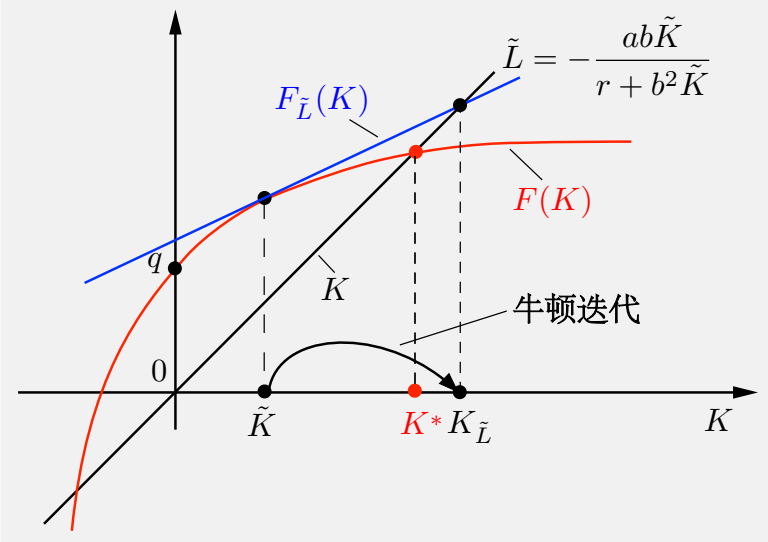


图 1.5.7: 线性二次型问题中采用一步前瞻的值空间近似方法。给定终止费用函数近似 $\tilde{J}(x) = \tilde{K}x^2$，我们计算相应的线性策略 $\tilde{\mu}(x) = \tilde{L}x$，其中

$$\tilde{L} = -\frac{ab\tilde{K}}{r + b^2\tilde{K}},$$

并利用图中所示的牛顿法的一步迭代得到相应的费用函数 $J_{\tilde{\mu}}(x) = K_{\tilde{L}}x^2$。

一维问题的黎卡提方程公式. 涉及最小化的黎卡提方程 *[参见式(1.41)和(1.42)]*

$$K = F(K), \qquad F(K) = \frac{a^2rK}{r + b^2K} + q.$$

对应于线性策略 $\mu(x) = Lx$ 的黎卡提方程

$$K = F_L(K), \qquad F_L(K) = (a + bL)^2K + q + bL.$$

稳定线性策略 $\mu(x) = Lx$ 的费用参数 K_L

$$K_L = \frac{q + rL^2}{1 - (a + bL)^2}.$$

当 K 处于稳定区时，相应的一步前瞻线性策略 μ_K 的线性参数 L_K *[参见式(1.48)]*

$$L_K = \arg\min_L F_L(K) = -\frac{abK}{r + b^2K}.$$

当 K 处于稳定区时，相应的一步前瞻线性策略 μ_K 的二次型费用参数 $\tilde{K}$ 通过求解线性的黎卡提方程

$$\tilde{K} = F_{L_K}(\tilde{K})$$

或者等价地，通过从 K 出发的牛顿迭代得到。

多步前瞻

对于多步前瞻最小化，同样可以得到类似牛顿迭代的解释。我们不在 $\tilde{K}$ 处对 F 进行线性化，而是在

$$K_{\ell-1} = F^{\ell-1}(\tilde{K})$$

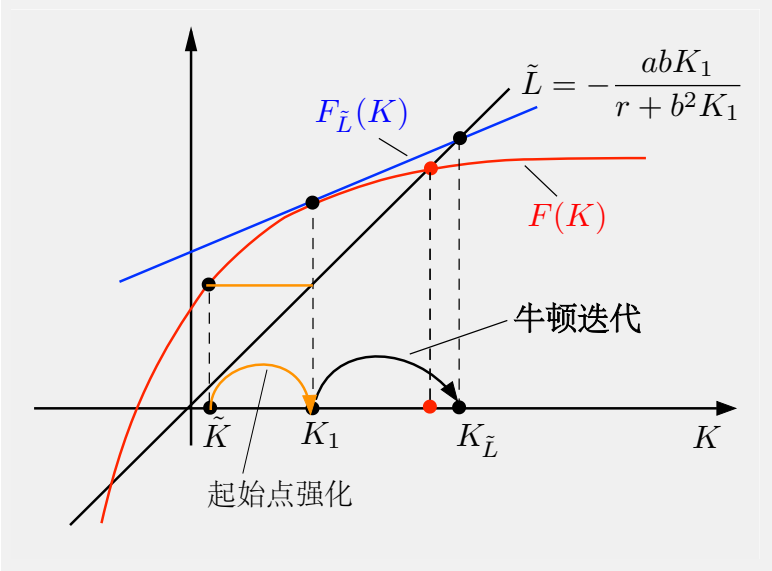


图 1.5.8: 图示了线性二次型问题中采用两步前瞻的值空间近似方法。以终止费用函数近似 $\tilde{J}(x) = \tilde{K}x^2$ 为起点，我们通过一次值迭代得到 K_1，从而改善了牛顿迭代的起始点。随后，我们利用所示的牛顿迭代计算出相应的线性策略 $\tilde{\mu}(x) = \tilde{L}x$，其中

$$\tilde{L} = -\frac{abK_1}{r + b^2K_1},$$

以及对应的费用函数 $K_{\tilde{L}}x^2$。由图可知，对于任意 $\tilde{K} \geqslant 0$，只要 ℓ 大于某个阈值，对应的 ℓ 步前瞻策略都将是稳定的。

处进行线性化，也就是说，线性化的点是从 $\tilde{K}$ 开始连续应用 $F^{\ell-1}$ 次所得到的结果。每一次 F 的应用都对应一次值迭代，因此牛顿迭代的等效起点为 $F^{\ell-1}(\tilde{K})$ *[the effective starting point for the Newton step is* $F^{\ell-1}(\tilde{K})$*]*。图 1.5.8 展示了当 $\ell = 2$ 时的情况。

稳定区域

接下来，我们将引入稳定区域（*region of stability*）这一概念，即满足

$$|a + bL_K| < 1$$

的所有 $K \geqslant 0$ 的集合，其中 L_K 是对应于 K 的一步前瞻策略的线性系数 [参见式(1.48)]。稳定区域也可以看作牛顿法的收敛区域（*region of convergence of Newton' s method*）。也就是说，这是所有使得牛顿法在求解黎卡提方程 $K = F(K)$ 时渐近收敛到 K^* 并且具有二次收敛速度（在当 $K \to K^*$ 附近）的初始点集合。注意，对于本书中所讨论的一维问题，稳定区域是一个区间 (K_S, ∞)，并且区间边界由单点 K_S 决定，该点正是函数 F 的导数等于 1 之处；参见图1.5.9。

对于多维问题，稳定区域可能不再能轻易地表示出来。不过，通常而言，随着前瞻步数的增加，稳定区域会扩大（*he region of stability is enlarged as the length of the lookahead increases*）。

事实上，随着前展步数的增加，有效的起始点

$$F^{\ell-1}(\tilde{K})$$

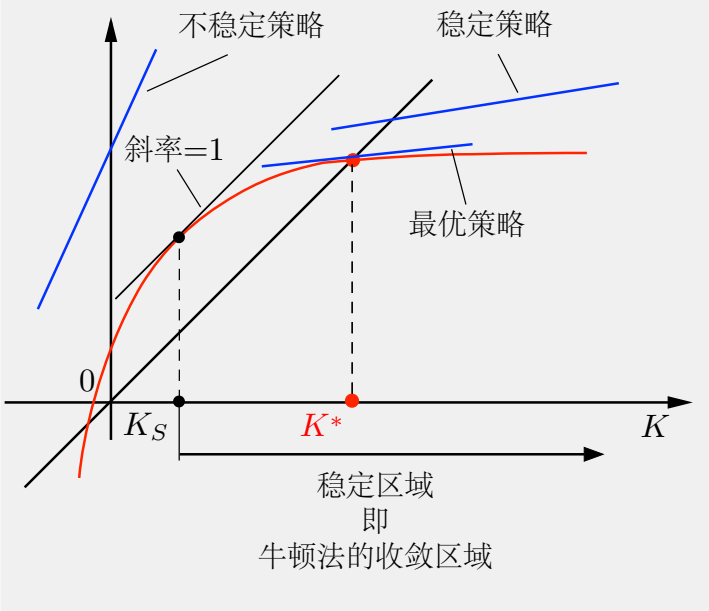


图 1.5.9: 图示了稳定区域，即所有使一步前瞻策略 μ_K 稳定的 $K \geqslant 0$ 构成的集合。这同时也是牛顿法能渐近收敛至 K^* 的初始点集合。

会越来越深入到稳定区域的内部。具体而言，对于任意给定的 $K \geqslant 0$，当前展步数 ℓ 超过某个阈值后，相应的 ℓ 步前展策略必然是稳定的（*for any given* $K \geq 0$, *the corresponding* ℓ*-step lookahead policy will be stable for all larger than some threshold*）；参见图1.5.8。专著 [Ber22b] 的第 3.3 节对稳定区域及多步前瞻对其扩大所起的作用有更加深入的讨论；另外可见习题 1.6 和 1.7。

一般无穷阶段问题中值空间近似的牛顿迭代解释

本节中我们讨论的将值空间近似解释为牛顿迭代，以及与此相关的稳定性概念，可以推广到本书及其他文献所讨论的一般的无穷阶段问题中。可以这么做的关键在于，我们的动态规划问题的理论框架允许状态空间和控制空间都是任意的，可以是离散的，也可以是连续的，甚至还可以进一步扩展到更为一般的抽象动态规划模型中；具体参见抽象动态规划专著 [Ber22a]。

在这一背景下，黎卡提算子被抽象的贝尔曼算子所取代。通过对贝尔曼方程、值迭代和策略迭代算法、一步和多步值空间近似、稳定区域以及特殊行为的图形化解释，可以获得许多宝贵的直观洞察；详尽讨论请参见专著 [Ber22b]。当然，这些图形化解释和可视化方法一般仅限于一维问题，但它们能够提供直观理解，激发对问题的猜测和进一步的数学分析，其中大量细节可参考专著 [Ber20b]。

多步前瞻中第一步的重要性

将值空间近似解释为牛顿步骤，能够深入理解在 ℓ 步前瞻中第一步的特殊性质。具体而言，仅仅第一步起到了牛顿迭代的作用（*it is only the first step that acts as the Newton step*），因此需要精确地实现（参见图1.5.5）。随后的 $\ell - 1$ 步前瞻则只是一系列从近似终止

费用函数 $\tilde{J}$ 开始的值迭代，它们的作用只是为了提高牛顿迭代的起始点的质量。因此，这些后续步骤的精确实现并不是至关重要的（*their precise implementation is not critical*），这也是作者在专著 [Ber22b] 中着重强调的一点。

这个思路表明，我们可以在合理范围内对多步前瞻中第一步以后的前瞻步进行简化，而且几乎不会带来明显的性能损失。其中一个重要的简化方式便是确定性等价，该方法将在后续不同章节的背景下加以讨论，例如见第 4.1.2 节。其他的可能简化方法还包括在第一步之后对前瞻树进行“修剪”；参见第 3.4 节。从实际应用角度出发，对第一步之后的多步前瞻进行适当简化可以极大减少在线计算量，而节省下来的计算资源则可以被有效地用来增加前瞻的步数，进而提高算法性能。大量的计算实验结果验证了相关简化方法的有效性，其中最早的工作始于 Bertsekas 和 Castanon 的研究 [BC99]，该文验证了在第一步之后使用确定性等价的有益效果。

1.5.2 策略前展与策略迭代

现在我们将从线性稳定的基本策略 μ 出发，讨论线性二次型问题的策略前展算法。该算法通过策略改进操作生成前展策略 $\tilde{\mu}$，根据定义，该操作产生对应于终止费用近似 $\tilde{J} = J_\mu$ 的一步前瞻策略。图1.5.10展示了策略前展算法的运作原理。如图所示，前展策略实质上是改进后的策略，其优越性体现在对所有状态 x 都满足 $J_{\tilde{\mu}}(x) \leqslant J_\mu(x)$。鉴于基本策略 μ 已被假设为稳定策略（即对所有 x 都满足 $J_\mu(x) < \infty$），这就意味着前展策略 $\tilde{\mu}$ 具有稳定性。

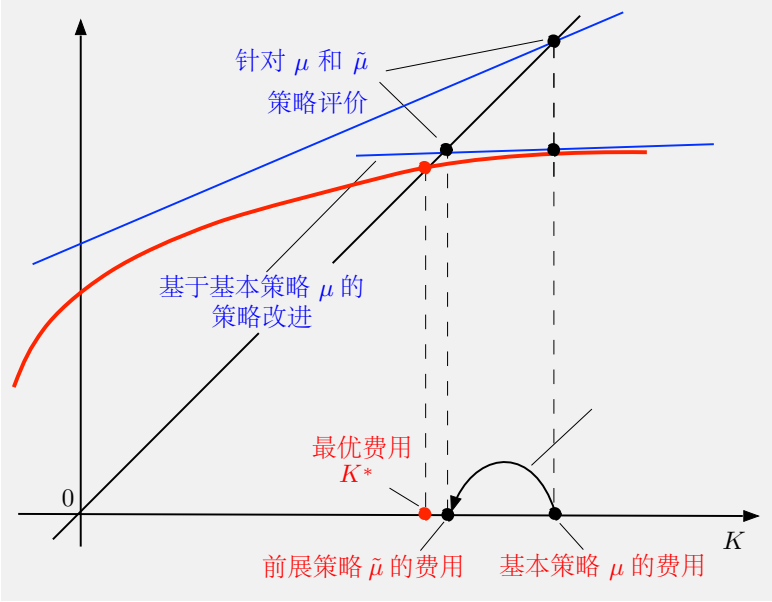


图 1.5.10: 线性二次型问题策略前展算法示意图。该算法从线性稳定基本策略 μ 出发，生成稳定的前展策略 $\tilde{\mu}$。前展策略 $\tilde{\mu}$ 的二次费用系数是以 μ 的二次费用系数为起点，通过将一步牛顿步迭代应用于黎卡提方程得到的。

由于前展策略本质上是一步前瞻策略，因此也可采用本节先前推导的公式进行描述。具体而言，设基本策略为

$$\mu^0(x) = L_0 x,$$

其中 L_0 为标量，且要求该基本策略必须稳定，即满足 $|a+bL_0|<1$。根据前文推导，μ^0 的费用函数的表达式为

$$J_{\mu^0}(x)=K_0x^2, \tag{1.54}$$

其中系数满足

$$K_0=\frac{q+rL_0^2}{1-(a+bL_0)^2}. \tag{1.55}$$

而前展策略 μ^1 则具有 $\mu^1(x)=L_1x$ 的形式，其系数为

$$L_1=-\frac{abK_0}{r+b^2K_0}; \tag{1.56}$$

参见公式(1.47)-(1.48)。

策略迭代算法本质上是非截短策略前展算法的循环执行，并经此生成稳定的线性策略序列 $\{\mu^k\}$。通过复现先前的推导过程可知，这些策略具有如下形式

$$\mu^k(x)=L_kx, \qquad k=0,1,\dots,$$

其中增益系数 L_k 通过以下迭代式生成：

$$L_{k+1}=-\frac{abK_k}{r+b^2K_k},$$

而系数 K_k 由下式给出：

$$K_k=\frac{q+rL_k^2}{1-(a+bL_k)^2},$$

[参见公式(1.55)-(1.56)]。

相应的费用函数序列为

$$J_{\mu^k}(x)=K_kx^2;$$

参见公式(1.54)。经典线性二次型理论表明，当初始策略为线性稳定策略时，J_{μ^k} 将收敛至最优费用函数 J^*，同时相应的线性策略序列 $\{\mu^k\}$（其中 $\mu^k(x)=L_kx$）将收敛至最优策略。如前述分析，系数序列 $\{K_k\}$ 具有二次收敛的特性。该结论最初由 Kleinman 在其著作 [Kle68] 中针对连续时间多维线性二次型问题完成证明，后续研究将其扩展至更广义问题范畴；详见著作 [Ber20b] 与 [Ber22b] 所列参考文献（Kleinman 将其结果的一维版本归功于贝尔曼与 Kalaba 的开创性工作 [BK65]）。

截短策略前展

一个基于稳定线性基本策略 $\mu(x)=Lx$、采用一步前瞻最小化以及终止费用近似 $\tilde{J}(x)=\tilde{K}x^2$ 的 m-步截短策略前展方法，其几何示意图如图1.5.11所示。截短前展策略 $\tilde{\mu}$ 是通过如下步骤计算得到的：从 $\tilde{K}$ 开始，利用 μ 执行 m 次值迭代，然后再执行一步用于求解黎卡提方程的牛顿迭代。

我们在第 1.1 节对截短策略前展的讨论中提到了一些有趣的性能问题。特别是我们指出：

(a) 通过策略前展实现的前瞻可能是通过直接最小化实现前瞻的一种经济替代方案：在显著降低计算成本的情况下，由此获得的前展策略的性能可能接近通过相同步数最小化计算得到的策略性能。。

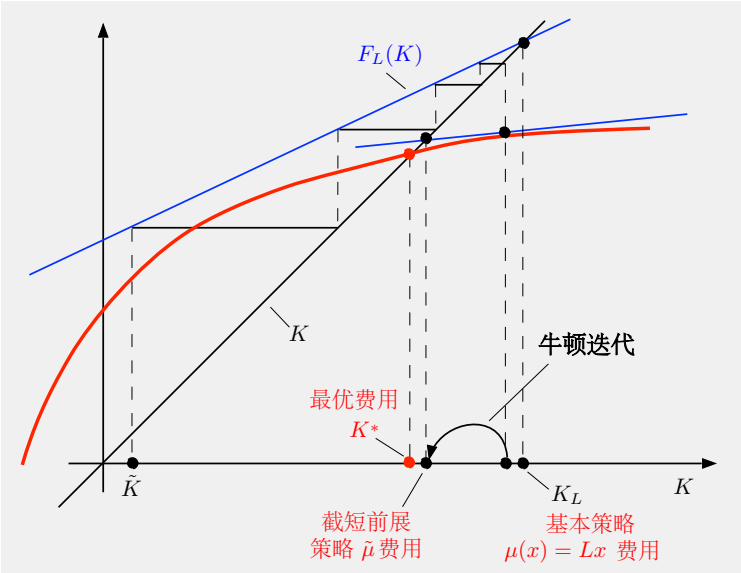


图 1.5.11: 截短策略前展的示意图：针对线性二次型问题，采用一步前瞻最小化、稳定的基本策略 $\mu(x) = Lx$，以及终止费用近似 $\tilde{K}$。在该图中，前展步数为 $m = 4$。

(b) 基于稳定策略的策略前展对由此得到的前展策略的稳定性特性具有积极影响。

要在一般情况下对这些结论进行严格的分析是困难的。然而，在一维线性二次型问题的背景下，可以借助类似图1.5.11的几何构造直观地理解这些结论。同时，这些结论也与计算实验的结果一致。更深入的讨论请参见专著 [Ber22b]。

两步牛顿迭代——基于值空间近似的策略前展

给定一个费用函数近似 $\tilde{K}$，我们可以得到相应的前瞻策略 $\tilde{\mu}(x) = L_{\tilde{K}}x$。在此基础上，我们可以考虑以 $\tilde{\mu}(x)$ 作为基本策略执行策略前展。这可以看作是在值空间近似基础上的策略前展，被称为*两步牛顿迭代 (double Newton step)*；即基于从 $\tilde{K}$ 出发的一步牛顿迭代的结果，继续执行一步牛顿迭代（见图1.5.12）。两步牛顿迭代远比基于 $\tilde{K}$ 的两步前瞻值空间近似更为强大，后者实际上等价于一次值迭代加一步牛顿迭代。此外，两步牛顿迭代的思想还可以推广到一般的无穷阶段问题中。

需要注意的是，上述基于值空间近似的策略前展算法也有多种变体，例如截短版、简化版和多步前瞻版。一个重要的截短版实例是 1996 年的 TD-Gammon 架构 [TG96]，其中基本策略本身就是通过值空间近似获得的，其相应的终止费用函数近似通过离线训练神经网络获得。

1.5.3 值空间近似的局部与全局误差界

在值空间近似中，一个重要的分析问题是如何量化通过一步或多步前瞻得到的策略的次优程度。这里的关键是理解从近似误差 $\tilde{J} - J^*$ 到性能误差 $J_{\tilde{\mu}} - J^*$ 的映射的性质（其中，$J_{\tilde{\mu}}$ 表示前瞻策略 $\tilde{\mu}$ 的费用函数，J^* 表示最优费用）。因此，我们将此映射称为关键映射。

在 α-折扣问题且状态空间 X 有限的情形下，存在一个经典的一步前瞻误差界，其形式

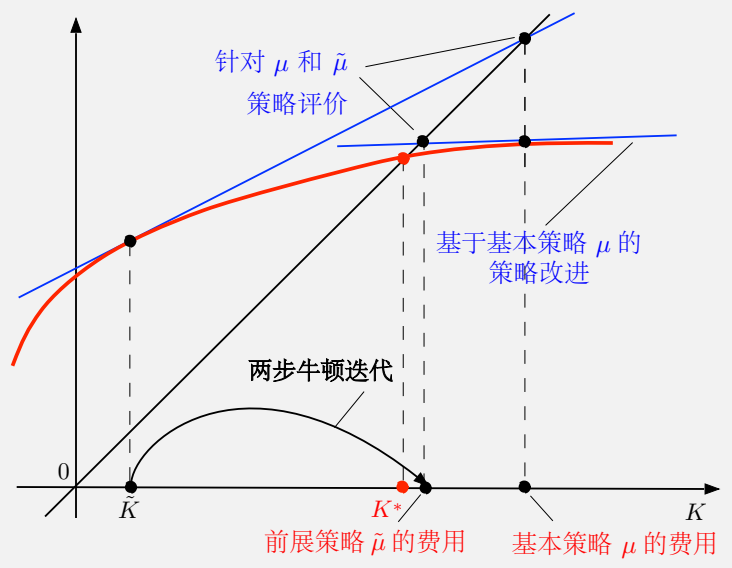


图 1.5.12: 两步牛顿迭代的示意图。在这里，基本策略 μ 是通过采用费用函数近似 $\tilde{J}(x) = \tilde{K}x^2$ 进行一步前瞻得到的。相应的前展策略 $\tilde{\mu}$ 的费用函数则是从 $\tilde{K}$ 出发，经过连续的两步牛顿迭代得到的。

为

$$\|J_{\tilde{\mu}} - J^*\| \leqslant \frac{2\alpha}{1-\alpha}\|\tilde{J} - J^*\|; \tag{1.57}$$

其中 $\|\cdot\|$ 表示最大范数，

$$\|J_{\tilde{\mu}} - J^*\| = \max_{x\in X}|J_{\tilde{\mu}}(x) - J^*(x)|, \qquad \|\tilde{J} - J^*\| = \max_{x\in X}|\tilde{J}(x) - J^*(x)|;$$

见附录 C。误差界(1.57)预测近似误差 $\|\tilde{J} - J^*\|$ 与性能误差 $\|J_{\tilde{\mu}} - J^*\|$ 之间存在线性关系。更一般化地，可以将 ℓ-步前瞻视为一种一步前瞻，只不过终止费用函数换为 $T^{\ell-1}\tilde{J}$，即对 $\tilde{J}$ 执行 $\ell - 1$ 次值迭代得到。由此可以得到 ℓ-步误差界：

$$\|J_{\tilde{\mu}} - J^*\| \leqslant \frac{2\alpha^{\ell}}{1-\alpha}\|\tilde{J} - J^*\|. \tag{1.58}$$

线性误差界(1.57)–(1.58)在图 1.5.13 中有所说明，不仅仅适用于 α-折扣问题，也适用于贝尔曼方程在函数子集上压缩映射的问题；参见强化学习专著 [Ber19b] 第 5.9.1 节，或抽象动态规划专著 [Ber22a] 第 2.2 节。

不幸的是，线性误差界非常保守，甚至在定性层面上也无法反映实际情况（*the linear error bounds are very conservative, and do not reflect practical reality, even qualitatively so*）。主要原因在于它们是全局误差界（*global error bounds*），即对所有 $\tilde{J}$ 都成立，包括最糟糕的情况。而在实际中，$\tilde{J}$ 往往被选择得足够接近 J^*，此时误差 $J_{\tilde{\mu}} - J^*$ 的表现与从 $\tilde{J}$ 出发的牛顿迭代的超线性收敛速率是一致的。换句话说，当 $\tilde{J}$ 相对接近 J^* 时，我们有如下局部（*local*）估计：

$$\|J_{\tilde{\mu}} - J^*\| = o\big(\|\tilde{J} - J^*\|\big), \tag{1.59}$$

其示意图见图1.5.14。

这种超线性关系的一个显著特征是：在牛顿法超线性收敛区域之外，性能误差会迅速上升。需要注意的是，对 $\tilde{J}$ 质量的小幅改进（例如在不改变近似结构的前提下，更好的采

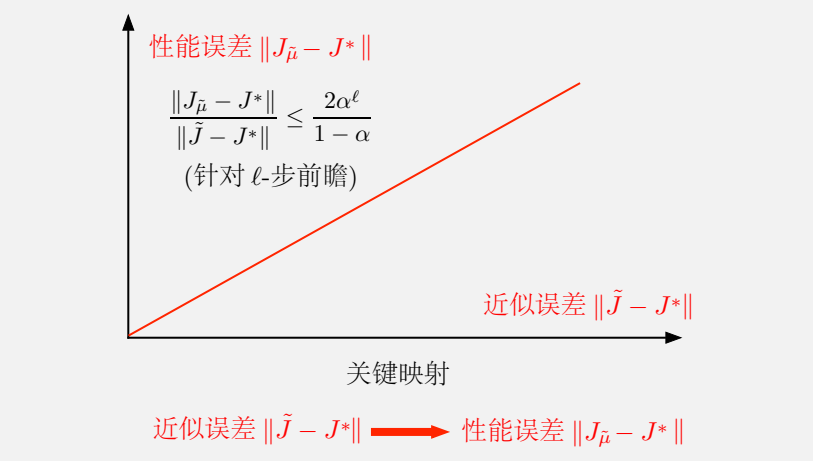


图 1.5.13: 采用 ℓ-步前瞻的值空间近似的误差界示意图。当 $\ell = 1$ 时，我们得到的一步前瞻的误差界即为式 (1.57)。

样方法、更精确的置信区间等）对所得策略的性能几乎没有影响——无论是在收敛区域内还是区域外（*little effect, both inside and outside the region of convergence*）。

在实际应用中，线性误差界(1.57)–(1.58)在数量和性质上往往存在巨大的差别 *[there is often a huge difference, both quantitative and qualitative, between the linear error bounds* (1.57)-(1.58) *and the superlinear error bound* (1.59)*]*。此外，线性误差界尽管在学术界颇受欢迎，却常常误导学术研究并让实践者困惑。[26]需要注意的是，正如我们先前提到的，图1.5.14所预测的性能表现趋势在值空间近似中具有非常广泛的适用性，因为它依赖于抽象动态规划的概念，而这些概念具有高度普适性，适用于任意状态空间、控制空间及其他问题特性；参见抽象动态规划著作 [Ber22a]。然而，需要指出的是，实现稳定性所需的前瞻长度在很大程度上取决于费用函数近似 $\tilde{J}$。

接下来，我们通过图1.5.15所示的一个例子，说明线性误差界(1.57)在预测一步前瞻策略性能时的失效情况。

例 1.5.1 (某线性二次型问题的全局与实际误差界). 考虑一维线性二次型问题，其系统为 $x_{k+1} = ax_k + bu_k$，每阶段的费用为 $qx_k^2 + ru_k^2$。我们考虑一步前瞻，并采用二次型费用函数近似

$$\tilde{J}(x) = \tilde{K}x^2,$$

其中 $\tilde{K}$ 属于位于稳定区域内，该区域为某个形如 (S, ∞) 的区间。本问题的黎卡提算子的定义为

$$F(K) = \frac{a^2 rK}{r + b^2 K} + q,$$

[26]Laidlaw、Russell 和 Dragan [LRD23] 的一项研究评估了当下流行的多种方法在 155 个问题上的实际表现，发现其实际表现与理论预测存在显著差异。引用该论文中的话：“我们发现现有的误差界所预测的性能表现与深度强化学习是否取得成功并不是很相关。”

该研究进一步指出，在值空间近似方法中，多步前瞻的步数（ℓ 的取值）较大对于保证稳定性具有重要意义。它还通过计算验证了一个已知的理论结果，即当前瞻长度 ℓ 足够大时，ℓ 步前瞻策略 $\tilde{\mu}$ 是严格最优的（但所需的 ℓ 长度取决于近似误差 $\tilde{J} - J^*$）。这一事实自 20 世纪六七十年代起就已为人所知，最初是在涉及 α-折扣的有限状态问题中证明的。该结果的推广形式可见于抽象动态规划著作 [Ber22a] 的命题 2.3.1；另见 [Ber22b] 一书的附录 A.4，该部分讨论了涉及不可微映射（例如贝尔曼算子）的方程组中牛顿法的收敛性。

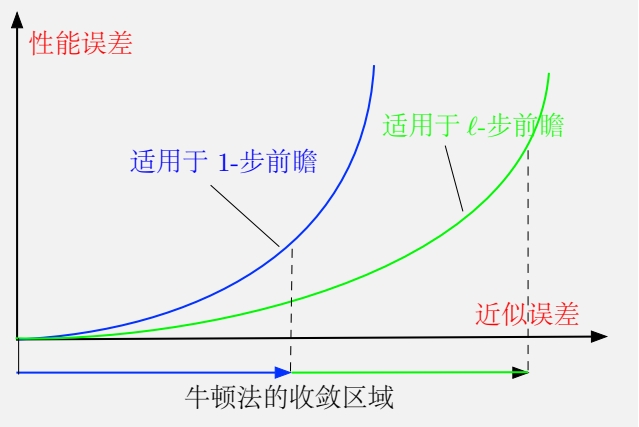


图 1.5.14: 值空间近似中的 ℓ-步前瞻近似方法的正确超线性误差界(1.59)的示意图。可以看到，在牛顿法的收敛区域之外，性能误差会迅速上升。[图中的示意并不完全真实；实际上收敛区域并非有界，因为它包含了形如 γe 的直线，其中 γ 为标量，e 为单位向量（所有分量均为 1）。] 需要注意的是，随着前瞻步数 ℓ 的增加，该收敛区域会扩大。此外，当前瞻步数 ℓ 足够大时，可以证明 ℓ-步前瞻策略 $\tilde{\mu}$ 在许多我们感兴趣的问题上都是精确最优的。这是一个理论结果，适用于 α-折扣有限状态问题等，并且自 20 世纪 60–70 年代就已为人所知（参见 [Ber22a] 的命题 2.3.1，其中证明了这一结果的更一般形式，适用范围超越折扣问题）。

一步前瞻策略 $\tilde{\mu}$ 的费用函数为

$$J_{\tilde{\mu}}(x) = K_{\tilde{\mu}} x^2,$$

其中 $K_{\tilde{\mu}}$ 通过对黎卡提方程 $K = F(K)$ 从初始值 $K = \tilde{K}$ 开始执行一步牛顿迭代得到。

设 S 为稳定区域的边界，即使黎卡提算子 F 对 K 的导数等于 1 的点：

$$\frac{\partial F(K)}{\partial K}\bigg|_{K=S} = 1.$$

则对于任意 $\overline{S} > S$，黎卡提算子 F 在区间 $[\overline{S}, \infty)$ 内均为是压缩映射，其压缩模量 α 取决于 $\overline{S}$ 的取值。具体来说，

$$\alpha = \frac{\partial F(K)}{\partial K}\bigg|_{K=\overline{S}},$$

并且由于 $\overline{S} > S$，且 F 的导数为正并随着 K 增大单调递减趋于 0，因此 $0 < \alpha < 1$。

对于二次型函数情形，可以重新推导误差界(1.57)，并将其用二次成本系数的形式改写为：

$$K_{\tilde{\mu}} - K^* \leqslant \frac{2\alpha}{1-\alpha}|\tilde{K} - K^*|, \tag{1.60}$$

其中 $K_{\tilde{\mu}}$ 是前瞻策略 $\tilde{\mu}$ 的二次费用系数（同时也是从 $\tilde{K}$ 出发执行一步牛顿迭代以求解不动点方程 $K = F(K)$ 所得到的结果）。图*1.5.15*展示了 $K_{\tilde{\mu}} - K^*$ 随 $\tilde{K}$ 变化的曲线，并与等式右侧的界进行了比较。可以看到，$(K_{\tilde{\mu}} - K^*)$ 的变化趋势体现了牛顿法的典型性质，与误差界 (1.60)的预测的情况截然不同。一个有趣的现象是，误差界(1.60)依赖于 α，而 α 又取决于 $\tilde{K}$ 与稳定区域边界 S 的接近程度；然而，牛顿法的局部收敛性质却与 S 无关。

1.5.4 值空间近似可能失效的原因及应对方法

最后，我们来讨论价值空间近似最常见的失效方式。考虑这样一种情况：终止费用近似 $\tilde{J}$ 是通过某种近似结构（如神经网络）利用训练数据获得的（例如在阿尔法零和 TD-Gammon

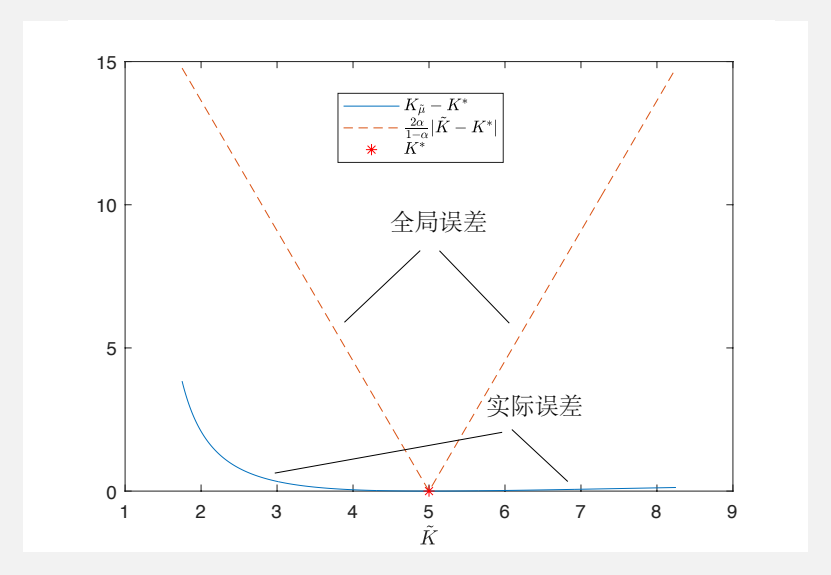


图 1.5.15: 一步前瞻误差 $K_{\tilde{\mu}} - K^*$ 随 $\tilde{K}$ 变化的全局误差界示意图，并与从 $\tilde{K}$ 出发执行一步牛顿迭代所得的真实误差进行比较；参见示例1.5.1。
图中问题的参数取值为为 $a = 2$、$b = 2$、$q = 1$ 以及 $r = 5$。在这些数值下，得到 $K^* = 5$，稳定区域为 (S, ∞)，其中 $S = 1.25$。图中使用的压缩模量 α 是在 $\overline{S} = S + 0.5$ 处计算的。根据所选的 $\overline{S}$ 值，α 可以任意接近 1，但随着 $\overline{S}$ 的增大而减小。需要注意的是，当 $\tilde{K} > K^*$ 时，误差 $K_{\tilde{\mu}} - K^*$ 远小于 $\tilde{K} < K^*$ 的情况，因为 F 的斜率随着 K 的增大而减小。这一点在全局误差界中并未体现出来。

中)。在这种情况下，决定近似误差 $\tilde{J} - J^*$ 的因素有三个组成部分：

(a) 近似结构的表达能力 (*power of the architecture*)，粗略来说，它衡量的是表示误差 (*representation error*)，即假设拥有无限的数据并以最优方式训练得到 $\tilde{J}$，仍然会存在的误差。

(b) 由于训练数据数量有限而导致的误差劣化 (*the error degradation due the limited availability of training data*)。

(c) 由于训练方法本身的不完善而导致的额外误差劣化 (*the additional error degradation due to imperfections in the training methodology*)。

因此，如果近似结构的表达能力不足，无法使 $\tilde{J} - J^*$ 落入牛顿法的收敛区域内，那么无论收集了多少数据、训练方法多么有效，基于一步前瞻的值空间近似都很可能会失败 (*Thus if the architecture is not powerful enough to bring $\tilde{J} - J^*$ within the region of convergence of Newton's method, approximation in value space with one-step lookahead will likely fail, no matter how much data is collected and how effective the associated training method is*)。

在这种情况下，有两种潜在的实际补救方法：

(1) 使用更强大的结构或神经网络来表示 $\tilde{J}$。

(2) 延长前瞻最小化与截短策略前展的总长度，以使 $\tilde{J}$ 的有效值落入牛顿法的收敛区域内。

第一种方法通常需要使用更深的神经网络或 Transformer，这意味着更多的权重参数以及更高的训练成本（参见第 5 章）。[27]第二种方法则需要更长的在线计算和/或仿真时间，这可能受到实际实时实现的限制。并行计算以及复杂的多步前瞻实现方法可能有助于缓解这些需求（参见第 3 章）。

1.6 强化学习与决策/控制

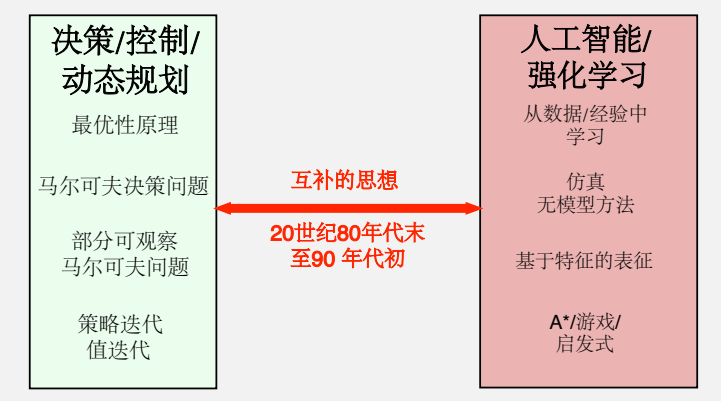


图 1.6.1: 决策与控制领域和人工智能领域之间思想协同作用的示意图。

强化学习当前的发展状况极大地受益于决策与控制领域和人工智能领域之间思想的交叉融合（见图 1.6.1）。这两个领域之间的紧密联系如今已被广泛认可。然而，二者之间仍存在文化上的差异。例如，决策与控制领域传统上依赖数学分析，而人工智能领域则更强调解决具有挑战性的实际问题。此外，在以强化学习为基础的讨论（使用人工智能相关术语）与以动态规划为基础的讨论（使用最优控制相关术语）之间，语言和关注重点上仍存在显著差异。

1.6.1 术语差异

本教材采用了动态规划与最优控制领域的专业术语。鉴于一些读者可能已经习惯于人工智能领域，或者最优控制领域的相关表述，我们在此提供一份这两个领域常用术语的对应列表，以便于读者理解。列表左侧为强化学习领域专用词汇，右侧为最优控制领域相对应的术语。

(a) 环境（*environment*）= 系统（system）。

(b) 智能体（*agent*）= 决策者（decision maker）或控制器（controller）。

(c) 动作（*action*）= 决策（decision）或控制（control）。

(d) 一个阶段的收益（*reward of a stage*）= 一个阶段的费用（cost of a stage）(的相反数)。

[27]有关使用一步前瞻和超大规模（2.7 亿参数）神经网络位置评价器实现特级大师级国际象棋程序的最新实例，参见 Ruoss 等人近期的工作 [RDM+24]。

(e) 状态价值（*state value*）= 从一个状态出发所花费用（cost starting from a state）(的相反数)。

(f) 价值函数或收益函数（*value function or reward function*）= 费用函数（cost function）(的相反数)。

(g) 最大化价值函数（*maximizing the value function*）= 最小化费用函数（minimizing the cost function）。

(h) 动作价值（*action value*）或状态动作价值（*state-action value*）= 一个状态与动作对的 Q-因子或 Q-值（Q-factor or Q-value of a state-control pair）。(Q-值也常用于强化学习的文献中。)

(i) 规划（*planning*）= 在数学模型已知的情况下，利用该模型来求解相应的动态规划问题（solving a DP problem with a known mathematical model）。

(j) 学习（*learning*）= 在不直接利用数学模型的情况下求解一个动态规划问题（solving a DP problem without using an explicit mathematical model）。(这是“学习”这一术语在强化学习文献中最常代表的含义。其他用法也很常见。)

(k) 自主学习（*self-learning*）= 通过某种形式的策略迭代来解决一个动态规划问题（solving a DP problem using some form of policy iteration）。

(l) 深度强化学习（*deep reinforcement learning*）= 采用深度神经网络来进行价值和/或策略近似的近似动态规划（approximate DP using value and/or policy approximation with deep neural networks）。

(m) 预测（*prediction*）= 策略评价（policy evaluation）。

(n) 广义策略迭代（*generalized policy iteration*）= 乐观策略迭代（optimistic policy iteration）。

(o) 状态抽象（*state abstraction*）= 状态聚集（state aggregation）。

(p) 时序抽象（*temporal abstraction*）= 时间聚集（time aggregation）。

(q) 学习一个模型（*learning a model*）= 系统辨识（system identification）。

(r) 分幕式任务（*episodic task*）或分幕（*episode*）= 有限步系统轨迹（finite-step system trajectory）。

(s) 持续性任务（*continuing task* ）= 无穷步系统轨迹（infinite-step system trajectory）。

(t) 经验回放（*experience replay* ）= 重复使用在一个仿真过程中生成的样本（reuse of samples in a simulation process）。

(u) 贝尔曼算子（*Bellman operator* ）= 动态规划映射（DP mapping）或动态规划算子（DP operator）。

(v) 回溯（*backup*）= 对一些状态求解其动态规划映射的值（applying the DP operator at some state）。

(w) 遍历（*sweep*）= 对所有状态求解其动态规划映射的值（applying the DP operator at all state）。

(x) 针对费用函数 J 的贪婪策略（*greedy policy with respect to a cost function* J）= 在由方程 J 定义的动态规划表达式中，取得最小值的策略（minimizing policy in the DP expression defined by J）。

(y) 后位状态（*afterstate*）= 决策后状态（post-decision state）。

(z) 实测真实值（*ground truth*）= 经验所得的证据（empirical evidence），或由直接观察所获得的信息（information provided by direct observation）。

上述的一些概念将会在后续章节中逐步引入；另见强化学习教材 [Ber19b]。因此，在后续学习中，读者可借助于以上列表，将本书所讲解的内容与其他强化学习文献中的相关知识串联起来。

1.6.2 符号差异

令人遗憾的是，除了所用术语不同，人工智能和最优控制领域所用的数学符号也不同。这也使得读者在参阅相关文献时，由术语不同而引起的误解进一步加剧。本教材基本采用了 Bellman 和 Pontryagin 在研究最优控制时所采用的“标准”数学符号，参见由 Athans 和 Falb 所著的 [AF66]，Bellman 所著的 [Bel67]，以及由 Bryson 和 Ho 所著的 [BH75] 等经典书籍。该系列符号也用于笔者所著的其他动态规划教材和学术专著中。

现将本教材中最常用的数学符号总结如下：

(a) x：状态。

(b) u：控制。

(c) J：费用函数。

(d) g：每阶段费用。

(e) f：系统方程。

(f) i：离散状态。

(g) $p_{xy}(u)$：在当前状态为 x，采用控制 u 的情况下，下一阶段状态为 y 的转移概率。

(h) α：折扣问题中的折扣率。

本教材所采用的 x-u-J 系列符号是最优控制教材采用的标准符号（例如，由 Athans 和 Falb 所著的 [AF66]，和由 Bryson 和 Ho 所著的 [BH75]，以及近来由 Liberzon 所著的 [Lib11]）。f 与 g 也是最优控制领域早期以及后续研究中最常用的代表系统和每阶段费用函数的符号（鉴于每阶段费用的英文是 cost，符号“c”相比“g”应当是更加自然的选择。然而遗憾的是，用符号“c”代表每阶段费用函数并不常见）。

符号 i（状态）和 $p_{ij}(u)$（转移概率）是离散系统马尔可夫决策问题和运筹学文献中常用的数学符号。有些文献用 $p(j \mid i, u)$ 代替 $p_{xy}(u)$ 来代表转移概率。

强化学习领域的相关文献绝大多数都是考虑有限多状态的马尔可夫决策问题，而且其中最常见的即是本书第 4 章中讲解的有折扣的和随机最短路径的无穷阶段问题。在这些文献中，最常见的符号即采用 s 代表状态，a 代表动作，$r(s,a,s')$ 代表每阶段收益，$p(s' \mid s,a)$ 或 $P_{s,a}(s')$ 代表在当前状态为 s 且采用动作 a 时，下一阶段状态为 s' 的转移概率，γ 代表折扣率。虽然这种符号体系非常适合有限状态问题，但对于连续空间模型却并不理想。原因在于它需要在连续空间上定义转移概率分布，从而使数学表达更复杂且不够直观。此外，对于不含随机成分的确定性问题，转移概率的记号显得既繁琐又没有必要。

1.6.3 动态规划与强化学习之间的关系

在比较强化学习与动态规划的方法论时，必须认识到两者在本质上通过其对序贯决策问题的共同关注而紧密相连。因此，任何可以通过动态规划解决的问题，原则上也可以通过强化学习来解决，反之亦然。

可以认为，强化学习的方法论范围比动态规划更广。它包括梯度下降与随机搜索算法、基于仿真的方法、采样与性能评估的统计方法，以及神经网络的设计与训练思想。然而，这类方法在与动态规划相关的研究和应用中也已被探讨多年，只是研究的深度和广度相对较小。

在人工智能对强化学习的理解中，机器通过与环境的交互，在试错过程中进行学习。[28]从实际角度看，这与动态规划所要实现的目标基本相同，但在强化学习中，通常更强调环境中的不确定性以及探索的过程。在决策、控制与优化领域，人们对利用强化学习方法解决难以处理的问题抱有浓厚兴趣，其中包括确定性的离散或整数优化问题——这些问题往往并不涉及数据收集、环境交互、不确定性或学习过程（实际上，自适应控制是决策与控制领域中唯一一个不确定性与探索起重要作用的问题类型）。

在应用层面上，动态规划最初于 20 世纪 50 至 60 年代发展起来，作为当时新兴的运筹学和最优控制方法的一部分。如今，这些方法已相当成熟，并为诸如机器人、自主交通、航空航天等领域提供了重要的工具、理论视角和丰富的应用，这些领域同样能够从强化学习的应用中获益。此外，动态规划已被广泛应用于工业工程、运筹学、经济学和金融等领域，因此这些领域的应用也可以从强化学习方法和思想的引入中受益。

与此同时，强化学习和机器学习为动态规划技术在新的领域中应用带来了机遇，例如机器翻译、图像识别、知识表示、数据库组织、大语言模型以及自动化规划等——在这些领域中，动态规划技术可能产生重大的实际影响。此外，可以说强化学习为序贯决策领域带来了全新的、富有雄心的精神，使得一些过去被认为远超动态规划能力范围的问题得以解决。事实上，在强化学习与动态规划之间的联系被认识到之前，诸如涉及中等维度欧几里得状态空间的问题，或部分可观察马尔可夫决策过程等高维问题，都被认为是动态规划方法完全无法处理的。

[28] 一种常见的描述是：“机器通过与环境交互所获得的反馈，逐步学习如何做出能够最大化奖励信号的决策。”

1.6.4 大语言模型与动态规划 / 强化学习之间的协同作用

强化学习与大语言模型（large language models，简写作 LLMs）能否形成协同作用？这是一个重要的问题，因为这两种人工智能范式在方法论、目标和能力上各不相同。强化学习侧重于序贯决策，而大语言模型则专长于自然语言的理解与生成，包括计算机代码的生成。

具体来说，强化学习旨在为顺序控制任务优化策略，在需要自适应决策的应用中（如机器人控制和资源分配）表现突出。相较之下，大语言模型擅长处理和生成类人文本，使其能够执行诸如翻译、摘要、情感分析和代码生成等任务。此外，凭借大量的预训练知识，大语言模型还能够在多样化的情境中实现泛化。

尽管二者存在差异，但强化学习与大语言模型的能力是*互补的*（*complementary*）——当二者结合使用时，可以成为极具威力的工具。下面我们总结强化学习与大语言模型在实践中可能实现协同作用的几种方式。

预训练 Transformer（如 ChatGPT）的出现，彻底革新了自然语言处理领域。这些模型可以通过离线训练进一步微调，以专注于特定任务或减轻不良偏差。值得注意的是，强化学习的方法，特别是策略空间近似技术，在这一微调过程中发挥了关键作用（详见第 5.5 节）。因此，强化学习通过离线优化技术在提升大语言模型能力方面起到了重要推动作用。

反过来，大语言模型也为强化学习提供了强大的推动力：它们能够注入领域知识、提升可解释性，并实现更符合人类思维方式的训练。通过利用自然语言输入，大语言模型有助于设计出更加透明且灵活的强化学习策略。此外，大语言模型还能通过辅助数学建模、算法选择和代码生成等方式支持强化学习的应用。这两者之间的互动仍在不断发展，推动着人工智能领域的持续创新。

总之，强化学习与大语言模型之间的结合不仅是“相加”的关系，而是“相乘”的关系：强化学习赋予大语言模型通过交互与反馈进行学习的能力，而大语言模型则为强化学习带来了上下文理解、可解释性、可访问性以及代码生成能力。两者结合，为更符合人类意图、能够交流并解释自身推理过程的高级应用铺平了道路。

1.6.5 机器学习与优化

机器学习与优化是紧密相互交织的领域，二者共享相似的数学模型和计算算法。[29]然而，它们在学术文化和应用背景上存在差异，因此值得对它们的相似性与差异性进行思考。

机器学习大体上可以分为三类主要方法，这三类方法都以某种形式涉及数据的收集与利用：

(a) 监督学习（*supervised learning*）：在这种方法中，首先收集包含大量输入–输出对的数据集（也称为带标签数据）。然后使用一种优化算法来构造一个参数化函数，使其既能很好地拟合这些数据，又能对新的、未见过的数据进行准确预测。监督学习问题通常被表述为优化问题，其示例将在第 5 章中介绍。一种常见的算法思路是使用基于梯度的算法最小化一个损失函数，该损失函数衡量数据集中实际输出与参数化模型预测输出之间的差异。

[29]这两个领域同样与统计分析领域密切相关。然而，在本节中我们不会重点讨论这种联系，因为它与本书的主要内容关联较小。

(b) 无监督学习（*unsupervised learning*）：在这种情况下，数据集是“无标签”的，也就是说，数据没有被划分为输入与对应输出的配对形式。无监督学习算法旨在发现数据中的模式或结构，这对于聚类、降维、密度估计等任务非常有用。其目标是从数据中提取有意义的信息与规律。某些无监督学习方法与动态规划存在一定联系，但这种联系并不紧密。总体而言，无监督学习与本书所关注的序贯决策类应用契合度不高。

(c) 强化学习（*reinforcement learning*）：强化学习与监督学习和无监督学习在一个关键方面有所不同——它并非以现成的数据集为出发点（*t does not use a dataset as a starting point*）。相反，它根据所采用的优化算法的需求（例如多步前瞻最小化、近似策略迭代与策略前展、或策略空间中的近似），在线或离线生成数据。[30]强化学习的进一步复杂之处在于：所生成的数据取决于用于控制系统的策略。理想情况下，数据应由最优或近似最优策略生成，但这种策略往往并不可得。因此，我们被迫使用一系列（希望是不断改进的）策略来收集数据——这正是动态规划（DP）中近似策略迭代算法的核心思想。这也是为什么该算法及其变体将成为本书讨论的重点之一。

另一类与动态规划和强化学习相关的机器学习方法是半监督学习（*semi-supervised Learning*）。这种方法利用同时包含带标签数据和无标签数据的数据集来训练模型。其思路是：从一部分带标签数据出发，逐步加入无标签数据，以构建一个“信息量丰富”的数据集，从而提升机器学习任务（如分类）的性能。这种方法介于监督学习（要求所有数据都有标签）与无监督学习（仅使用无标签数据）之间。半监督学习与主动学习（*active learning*）领域也密切相关，在主动学习中，使用类似动态规划的方法来逐步扩充带标签的数据集。有关更多讨论，可参考 Zhu 和 Goldberg 的专著 [ZG09]，Van Engelen 和 Hoos 的综述 [VEH20]，以及 Marchesoni-Acland 等人 [MAMKF23] 和 Bhusal、Miller、Merkurjev [BMM25] 的应用实例论文。

另一方面，优化问题及其算法可能涉及，也可能不涉及数据的收集与使用。只有在某些特定应用背景下（其中大多数与机器学习有关）时，数据才会被引入。从理论角度来看，优化问题通常根据其数学结构进行分类，而数学结构也是决定采用何种方法求解的主要依据。尤其常见的分类是区分静态优化问题（*static optimization problems*）与动态优化问题（*dynamic optimization problems*）：后者涉及序贯决策，并且不同决策之间存在反馈关系；而前者则只涉及单次决策。

具有完整或不完整状态观测的随机问题属于动态问题（除非它们仅涉及不带反馈的开环决策），这类问题的最优求解需要借助动态规划。确定性问题可以被表述为静态问题，但也常常出于算法便利的原因被表述为动态问题。在这种情况下，决策过程被（有时是人为地）划分为若干阶段——本书在讨论离散优化及其他情形时，也经常采用这种处理方式。

另一种重要的优化问题分类方式是根据其搜索空间是离散的（*discrete*）还是连续的（*continuous*）来划分。离散问题包括确定性问题，如整数优化和组合优化问题，可以通过整数规划的形式方法或动态规划来求解。这类问题通常具有较高的复杂性，因此往往需要借助启发式方法进行（次优）求解。

连续优化问题通常采用与离散问题截然不同的方法来求解，这些方法基于微积分与凸性理论，例如拉格朗日乘子法与对偶理论，以及线性、非线性和凸规划等计算方法。某些离

[30]强化学习的一种变体称为离线强化学习（*offline RL*）或批量强化学习（*batch RL*），它以历史数据集为起点，而不再通过与环境交互来收集新的数据。

散问题，尤其是涉及图结构的问题（如匹配、运输和转运问题），可以通过依赖线性规划与对偶理论的连续空间网络优化方法来解决。混合型问题（同时包含离散变量与连续变量）通常需要采用离散优化技术，但也可以从凸对偶方法（本质上属于连续方法）中获益。

总体而言，只要一个优化问题能够被表述为序贯决策问题（*as long as it is formulated as a sequential decision problem*）（如第 1.2–1.4 节所述），无论它是确定性的还是随机的、静态的还是动态的、离散的还是连续的，动态规划的方法论都可以加以应用。在算法结构上，动态规划与其他优化技术（尤其是基于微积分与凸性的方法）存在显著差异。值得注意的是，动态规划既能处理离散问题，也能处理连续问题，并且不受局部极小值的困扰——其目标始终是寻找全局最优解。

需要注意的是，优化与机器学习之间存在一种本质上的区别：优化主要围绕数学结构以及相应算法的基础性问题分析而展开；而机器学习则主要围绕数据的收集、使用与分析组织研究，且通常更加强调统计方面的问题。（*the former is mostly organized around mathematical structures and the analysis of the foundational issues of the corresponding algorithms, while the latter is mostly organized around how data is collected, used, and analyzed, often with a strong emphasis on statistical issues.*）这一重要差异深刻地影响了两大领域研究者的思维方式与研究视角。

1.6.6 机器学习与优化中的数学

现在我们来讨论优化领域与机器学习领域在研究文化上，特别是在数学运用方面的一些差异。在优化领域，研究重点通常放在具有普适性的算法上，这些方法能够针对各种类型的问题提供广泛且数学上严格的性能保证。普遍认为，为某种优化方法建立坚实的数学基础，不仅能增强其可靠性，还能明确其适用范围的边界。此外，人们也认识到，对所采用优化方法的数学结构有深入理解，能显著提升对实际问题的建模能力，并有助于选择合适的算法来匹配具体问题。

机器学习研究中也包含若干具有强烈数学特征的重要研究方向，特别是与理论计算机科学、复杂性理论和统计分析相关的部分。与此同时，机器学习中存在许多极具实用价值的算法结构——如神经网络、大语言模型以及图像生成模型——这些结构在数学上仍缺乏深入理解，并在很大程度上难以进行严格的数学分析。[31]这也导致一种看法：在许多实际的机器学习场景中，相较于专注于数学上的严谨分析，投入更多精力于实际实现与应用可能带来更高的收益。

此外，机器学习的研究往往以特定的数据集或专门类型的训练问题（例如语言翻译或图像识别）为出发点。其首要目标是找到一种在该特定数据集或问题上效果良好的方法，即使这种方法无法推广到其他场景也无妨。因此，在机器学习的语境中，那些针对特定问题或数据类型表现良好的专用近似结构、实现技术与启发式方法，即使缺乏严格或普适的性能保证，也完全可以被接受。

总之，优化与机器学习都在重要层面上依赖数学建模与严格分析，并且在所使用的技术与工具、以及所涉及的实际应用中存在大量重叠。然而，根据具体问题的不同，这两个领

[31]举例来说，He 等人的论文 “Deep Residual Learning for Image Recognition”，发表于 Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, 2016，截至 2025 年 6 月已被引用超过 272,218 次，而全文仅包含两个公式。著名的神经网络架构论文 Vaswani 等人的 “Attention is All You Need”，发表于 NIPS, 2017，为 GPT 奠定了基础，截至 2025 年 6 月已被引用超过 185,110 次，而全文仅包含六个公式。

域在数学分析、理论洞察与普适性，以及实际效果与问题特定效率之间所强调的重点往往不同。这种差异有时会导致一定的张力，因为不同领域的研究者未必能够充分理解或欣赏彼此的研究视角。

1.7 注释、资源与习题

我们现在对本章进行总结，并说明如何灵活地将其作为若干不同课程的基础。同时，我们还将对动态规划和强化学习的相关文献进行简要回顾，并给出一些曾在亚利桑那州立大学（ASU）课程中使用过的练习题。

本章总结

在本章中，我们旨在概述近似动态规划和强化学习的整体框架，为课堂上进一步深入探讨其他强化学习主题奠定基础。具体而言，我们以不同的深度介绍了以下内容：

(a) 精确动态规划的算法基础，涵盖其主要形式：确定性与随机性、离散与连续、有限阶段与无穷阶段。

(b) 值空间的近似：包括一步与多步前瞻方法。这是强化学习的核心技术，也是阿尔法零等主要成功案例的理论基础。我们对比了值空间近似与策略空间近似，并讨论了二者的结合方式。

(c) 离线训练与在线执行之间的重要区分：在值空间近似的背景下，我们强调了它们之间的协同关系，并指出这种协同可通过牛顿法直观地加以解释。

(d) 策略迭代与策略前展等基本方法：前者主要是一种离线方法，后者则是更灵活的在线方法。两种方法及其变体都与牛顿法密切相关，其有效性正源于这种联系。

(e) 函数近似的应用：这是本书中反复出现的主题之一。我们介绍了一些主要的近似结构，如神经网络和基于特征的体系结构。

一些具有广泛应用的重要模型——例如离散优化、部分可观察马尔可夫决策过程、多智能体问题、自适应控制以及模型预测控制——将会在下一章中讨论。本章的主要目标之一是为多种强化学习课程提供一个基础平台，以便在更深入的层面上探讨各种算法及其相应的理论，例如：

(1) 策略前展与策略迭代。

(2) 用于离线训练的神经网络及其他近似结构。

(3) 聚集方法，可用于值空间近似中的费用函数近似。

(4) 更广泛的序贯决策问题讨论，包括系统参数变化、顺序估计、以及系统辨识与控制的联合问题。

(5) 随机算法，如时序差分方法与 Q-学习，用于近似策略迭代框架下的离线策略评价。

(6) 采样方法，用于在费用函数与策略近似的背景下收集离线训练数据。

(7) 仿真型算法中的统计估计与采样效率改进，包括置信区间和计算复杂度的分析。

(8) 针对特殊结构问题的在线方法，例如多臂老虎机类型的问题。

(9) 基于仿真的策略空间近似算法，包括策略梯度与随机搜索方法。

(10) 控制系统设计方法的深入探讨，如模型预测控制与自适应控制，以及它们在机器人与自动化交通系统中的应用。

在我于亚利桑那州立大学开设的课程 (2019–2025) 中, 我重点讲授了上述方法中的 (1)–(4)，并对 (9) 做了有限的介绍。在其他课程中，可以基于本章及下一章的内容，从以上列表中选择不同的主题进行讲授。

各节注释与参考文献

在接下来的文献综述中，我们将主要关注教材、研究专著和综合性综述论文，这些资料用于补充本书的讨论、展示相关观点，并共同为读者提供一份文献导读。不可避免地，我们的选择带有一定的学术文化偏向，并对作者熟悉、且在风格上与本书一致的资料（包括作者本人的著作）给予了较多关注。我们在此提前致歉，因为这种取向可能导致遗漏一些超出作者理解范围或研究视角之外的文献参考。

第 1.1–1.4 节: 鉴于本教材着重讲解近似动态规划与强化学习，在本章中我们只简要的介绍了精确动态规划的一些相关知识点。笔者的动态规划教材 [Ber17] 采用了与本书一致的符号与写作风格，对精确有限阶段动态规划及其在离散与连续空间中的应用进行了全面的讲解。由 Putman 所著的 [Put94] 和由笔者所著的 [Ber12] 详尽地讲解了无穷阶段有限状态的马尔可夫问题。笔者的 [Ber12] 也包含了对于连续空间无穷阶段问题的讲解，其中包括本章所介绍的一维线性二次型问题。连续空间问题在分析与计算上具有特殊的挑战性，这正是当前强化学习方法研究的前沿方向之一。值得一提的是，笔者 1976 年的动态规划教材 [Ber76] 是首部在框架上允许任意状态、控制与扰动空间的动态规划专著。

关于精确动态规划中较为复杂的数学问题，Bertsekas 与 Shreve 的专著 [BS78] 进行了深入探讨，特别是针对随机最优控制中与概率论 / 测度论相关的问题，包括部分状态信息问题。该专著对这些议题进行了系统而详尽的论述。随后，Huizhen Yu 与笔者的后续工作 [YB15] 解决了与策略迭代相关的特殊的可测性问题，并进一步分析了值迭代的收敛性。作者的动态规划专著第二卷 [Ber12] 的附录 A 对 [BS78] 一书中的测度论框架给出了简明易懂的介绍。[32]

作者的抽象动态规划专著第三版 [Ber22a] 是对 2013 年第一版的扩展与深化，旨在统一地发展总费用型序贯决策问题的核心理论与算法。该书同时涵盖了随机型、极小极大型、

[32] 严格的随机最优控制数学理论（包括建立合适的 Borel 空间测度论框架）起源于 20 世纪 60 年代，由 Blackwell 的工作 [Bla65],[Bla67] 奠定基础。该理论依赖于描述集合论中的解析集理论——这一概念最早由俄罗斯青年数学家 M. Suslin 于 1917 年提出，并由其导师 N. Luzin 进一步发展。这一发展在 Bertsekas 与 Shreve 的专著 [BS78] 中达到顶峰，该书建立了如今的“标准”框架，基于 Borel 空间、下半解析函数以及普适可测策略的形式体系。这一理论的发展伴随着极其复杂的数学问题，其中一个关键来源在于：当一个关于两个变量 x 和 u 的 Borel 可测函数 $F(x,u)$ 对 u 取最小值时，所得到的函数 $G(x)=\min_u F(x,u)$ 不一定是 Borel 可测的（它属于更广泛的下半解析函数类）。这一关键事实正是 Suslin 分析的出发点。此外，即使最小值可由多个策略 μ 取得，即 $G(x)=F\big(x,\mu(x)\big)$ 对所有 x 都成立，也有可能这些 μ 中没有任何一个是 Borel 可测的（不过在此情况下，在更广义的“普适可测策略”类中，确实存在一个使最小值成立的策略）。因此，即便一开始假设费用函数和策略都是 Borel 可测的，在执行动态规划算法（如值迭代与策略迭代）时，也会很快跳出这一框架。为弥补这一缺陷，在没有额外（且相当强）的假设条件下，就必须采用 [BS78] 中提出的更广泛的普适可测性框架。

博弈型、风险敏感型及其他类型的动态规划问题，通过引入抽象动态规划算子（或称贝尔曼算子，本书采用此称呼）加以统一表述。其核心思想是通过抽象获得洞见。具体而言，一个动态规划模型的结构可以由其抽象贝尔曼算子编码，而该算子可被视为模型的“数学指纹”。因此，该算子的性质（如单调性与压缩性）在很大程度上决定了可以应用于该模型的分析结果与计算算法。此外，抽象动态规划的思想对于强化学习方法的可视化与解释也非常有用，特别是在使用牛顿法形式主义分析的情境下——本书在讨论线性二次型问题时对此已作简要介绍。

值空间近似、策略前展与策略迭代是本书的主要主题。[33]这些方法是极为强大且通用的技术：它们可应用于确定性与随机问题、有限与无穷阶段问题、以及离散与连续空间问题，乃至这些类型的混合形式。此外，策略前展方法具有可靠、易于实现的优点，并且可以与在线重新规划结合使用。同时，它还能与 Transformer 网络和大语言模型等新兴技术兼容（参见第 3.3.7 节）。

正如前文所述，基于给定基本策略的策略前展，实际上就是从该基本策略出发执行的策略迭代算法的第一次迭代。而截短策略前展可以理解为一种“乐观型”的单次策略迭代形式，即通过执行有限次值迭代来对策略进行近似评估。有关详细讨论，可参见文献 [Ber20b] 与 [Ber22b]。[34]

第 1.5 节: 关于线性二次型问题的文献极为丰富。针对此类问题，策略迭代与牛顿法之间的联系最早由 Kleinman [Kle68] 推导得出。该工作是他在 MIT 师从 M. Athans 期间的博士研究成果。Kleinman 的工作主要聚焦于连续时间线性二次型问题（关于离散时间情形，可参见 Hewer 的论文 [Hew71]）。与近似策略迭代相关的后续研究可参见 Feitzinger、Hylla 与 Sachs 所著的 [FHS09] 以及 Hylla 的博士论文 [Hyl11]。

值空间近似与牛顿法之间的普遍联系（超越策略迭代的范围），以及它与模型预测控制和自适应控制之间的关联，最早由笔者在著作 [Ber20b]、论文 [Ber21], [Ber22c]，以及专著 [Ber22b] 中提出并进行了深入讨论。这种联系为深入理解值空间近似体系结构中离线训练与在线执行之间的协同作用提供了起点，其中包括多步前瞻在改善牛顿法初始点方面所起的作用。此外，专著 [Ber22b]（以及本书的附录 B）还分析了用于不可微分不动点问题的牛顿法变体，例如在贝尔曼方程中出现的情况（其中右端项在有限控制空间问题中通常是不可微的）。

需要注意的是，在值空间近似中，我们实际上是将牛顿法应用于一个方程组（即贝尔曼方程）的求解。该应用与强化学习中常用于解决特定类型优化问题的梯度下降方法（例如神经网络训练问题，参见第 5 章）没有任何关联。特别地，并不存在能够用于求解像贝尔曼方

[33] “策略前展（英文为 rollout）”（英文也写作 “policy rollout”）这一名称最早由 Tesauro 和 Galperin 在论文 [TG96] 中提出，源自于他们研究的西洋双陆棋游戏中的“掷骰子（rolling the dice）”的过程。在 Tesauro 的设想中，一个给定的双陆棋局面可以通过从该位置开始模拟多局游戏直至结束来进行评估。引用 [TG96] 原文中的描述：“在双陆棋的术语中，一个局面的期望值被称为该局面的权益（equity），而通过蒙特卡罗采样估计这一权益的过程被称为执行一次策略前展（rollout）。这包括从该局面出发，在使用不同的随机骰子序列下，多次完整地进行游戏，并使用固定的策略为双方做出行动决策。”

[34] 截短策略前展（英文为 truncated rollout）最早也是由 Tesauro 和 Galperin 在 [TG96] 一文中提出。以下引用原文说明：“在实时博弈决策中，使用大型多层神经网络执行完整的策略前展是不可行的，因为这些大型网络的运算速度至少比前文所述的线性评价器慢 100 倍。因此，我们研究了一种替代的蒙特卡罗算法，即所谓的截短策略前展。在这种技术中，模拟试验不会执行到游戏结束，而是仅进行若干步仿真；然后使用神经网络对最终到达的局面进行权益估计，以替代真实的游戏结果。截短策略前展算法显著减少了 CPU 时间，主要有两个原因：首先，每次试验的步骤可能大幅减少；其次，每次试验的方差也显著降低，因为只进行少量随机步骤并记录一个实值估计，而非执行大量随机步骤并得到一个整数最终结果。这两个因素共同作用，使得截短策略前展的计算速度相比完整的策略前展提高了至少一个数量级，同时相对于基本策略仍能显著降低误差。”自 1996 年以来，对截短策略前展的分析与计算实验结果与上述评价高度一致。

程这类方程组的梯度下降方法。然而，确实存在一些可应用于具有特殊结构方程组（包括贝尔曼方程）的“一阶”确定性算法，如 Gauss-Seidel 方法和 Jacobi 方法（以及它们的随机异步类的扩展）。此类方法包括多种 Q-学习算法，这些算法已在 Bertsekas 与 Tsitsiklis 的《神经元动态规划》[BT96] 一书以及 Meyn 的最新著作 [Mey22] 中进行了讨论。虽然这些方法在某些情况下确实有用，但它们的收敛速度远慢于牛顿法，并且在在线执行的场景中，其应用价值有限。

强化学习文献

最早的动态规划 / 强化学习著作诞生于 20 世纪 90 年代，为该领域后续的发展奠定了基调。其中包括 Bertsekas 与 Tsitsiklis 于 1996 年出版的著作 [BT96]，该书体现了决策、控制与优化的视角；Sutton 与 Barto 于 1998 年出版的著作（第二版 [SB18] 于 2018 年出版），则体现了人工智能视角下的观点。关于与本书相关的若干主题（如算法收敛性问题及其他动态规划模型，包括基于平均费用的模型与半马尔可夫优化问题），可参考前者 [BT96] 以及作者的动态规划教材 [Ber12]，[Ber17] 中的更广泛讨论。需要注意的是，上述两部强化学习著作均针对有限状态马尔可夫决策模型，采用转移概率的符号体系；它们并未讨论连续空间问题，而这正是本书的主要研究重点之一。

近期出版的相关书籍包括由 Gosavi 所著的 [Gos15]（作为该专著第二版，极大地扩展了 2003 年问世的第一版的内容），着重介绍了基于仿真的优化与强化学习算法；Cao 所著的 [Cao07] 着眼于通过基于敏感度的方式来讲解基于仿真的方法；Chang、Fu、Hu 和 Marcus 所著的 [CHFM13]（其 2007 年专著的第二版）强调了有限阶段/有限前瞻方法及自适应采样；Busoniu、Babuska、De Schutter 与 Ernst 所著的 [BBDSE17] 重点介绍了适用于连续空间系统的函数近似方法，并且包含了关于随机搜索方法的讨论；Szepesvari 所著的简短的专著 [Sze10]，重点介绍若干主要的强化学习算法，如时序差分法、多臂老虎机方法与 Q-学习，Powell 所著的 [Pow11] 强调了在资源分配和运筹学领域的应用；Powell 与 Ryzhov 所著的 [PR12]，聚焦于学习与贝叶斯优化中的特殊主题；Vrabie、Vamvoudakis 和 Lewis 所著的 [VVL13] 讨论了基于神经网络的方法、在线自适应控制以及连续时间最优控制的应用；Kochenderfer 等所著的 [KAC+15] 有选择性地讨论了动态规划中的近似、应用实例和处理不确定性的方法；Jiang 与 Jiang 所著的 [JJ17] 在近似动态规划的框架内发展了自适应控制理论；Liu、Wei、Wang、Yang 与 Li 所著的 [LWW+17] 讨论了自适应动态规划的一些形式，以及强化学习和最优控制的某些话题；Zoppoli、Sanguineti、Gnecco 与 Parisini 所著的 [ZSGP20]，研究最优控制中的神经网络近似以及具有非经典信息结构的多智能体 / 团队问题；Meyn 的著作 [Mey22] 从数学视角出发，重点探讨了强化学习与最优控制之间的联系，并对随机问题与相关算法进行了更为详细的分析；Zhao 的 [Zha25] 以通俗易懂的表述介绍了强化学习中的数学理论。

还有一些著作，虽然并非专门聚焦于动态规划或强化学习，但也涉及本书讨论的若干主题。其中，Borkor 的书 [Bor08] 是适用于专业学者的学术专著。该书通过采用所谓的常微分方程的方式，严谨地讨论了近似动态规划中许多迭代随机算法的收敛性问题。Meyn 的 [Mey08] 包含了更为广泛的话题，但同时也涉及到了我们讨论的一些近似动态规划算法。Haykin 的 [Hay08] 在以神经网络为主题的更广阔的背景下探讨了近似动态规划。由 Krishnamurthy 所著的 [Kri16] 聚焦于讨论部分状态信息的问题，同时也包含了关于精确动态规划和近似动

态规划/强化学习方法的讨论。Kouvaritakis 与 Cannon 的 [KC16]、Borrelli、Bemporad 与 Morari 的 [BBM17]、以及 Rawlings、Mayne 与 Diehl 所著的 [RMD17]，这三本教材共同系统地阐述了模型预测控制方法论。Lattimore 与 Szepesvári 的 [LS20] 专注于多臂老虎机方法。Brandimarte 的 [Bra21] 是一本动态规划 / 强化学习的教学入门书，强调其在运筹学中的应用，并附带 MATLAB 代码。Hardt 与 Recht 的 [HR22] 聚焦于更广泛的机器学习主题，并选择性地涵盖了近似动态规划与强化学习相关内容。

本书在风格、术语和符号体系上与作者近年的几部教材保持一致，包括 [Ber19b] 《强化学习与最优控制》、[Ber20b] 《策略前展、策略迭代与分布式强化学习》、[Ber22b] 《阿尔法零对最优、模型预测和自适应控制的启示》以及《抽象动态规划》专著第三版 [Ber22a]。这些著作共同构成了对该领域较为系统、数学化的完整阐述。其中，[Ber19b] 一书对值空间近似方法进行了更广泛的讨论，包括确定性等价控制与聚集方法；同时，它还比本书更详细地探讨了策略空间近似。[Ber20b] 更专注于策略前展、策略迭代与多智能体问题，并首次提出了值空间近似与牛顿法之间的联系。[Ber22b] 则主要聚焦于这种联系，并基于 [Ber20a] 和 [Ber22c] 中的分析进行了系统阐述。[Ber22a]（2013 年首版的第三版）是一本关于精确动态规划的高阶专著，建立了以贝尔曼算子为核心的数学框架，这一框架是本书及 [Ber20b]、[Ber22b] 中基于牛顿法的可视化分析的核心基础。

除教材外，与本书主题相关的综述论文和简短研究专著也已大量出现，且数量仍在迅速增长。有很大影响力的早期综述包括由 Barto、Bradtke 和 Singh 所写的 [BBS91]（讨论了实时动态规划方法及其先例，实时启发式搜索 [Kor90]，以及异步动态规划理念 [Ber82]、[Ber83]、[BT89] 在相关背景下的应用），以及由 Kaelbling、Littman 和 Moore 所写的 [KLM96]（聚焦于强化学习的一般原则）。这些综述都从人工智能的视角出发讲解了相关的方法。由 White 和 Sofge 编辑的 [WS92] 也包含了描述该领域早期工作的一些综述。

由 Si、Barto、Powell 和 Wunsch 编辑的 [SBPW04] 中包含的一些概述文章描述了本书中没有详细讲解的方法：线性规划（De Farias 的 [DF04]）、大规模资源分配方法（Powell 和 Van Roy 的 [PVR04]）以及确定性最优控制方法（Ferrari 和 Stengel 的 [FS04]，以及 Si、Yang 和 Liu 的 [SYL04]）。更新的关于这些及相关话题的讨论则可见由 Lewis、Liu 和 Lendaris 编辑的概述文集 [LLL08] 以及由 Lewis 和 Liu 编辑的 [LL13]。

近年来，该领域的长篇综述和短篇专著包括由 Borkar 所著的 [Bor09]（从方法论的视角探讨了强化学习与其他蒙特卡罗方法的联系）、Lewis 和 Vrabie 的 [LV09]（展现了控制的视角）、Szepesvari 的 [Sze10]（从强化学习的视角探讨了值空间的近似）；Deisenroth、Neumann 和 Peters 所著的 [DNP13] 与 Grondman 等人所著的 [GBLB12]（专注于讨论策略迭代方法）；Browne 等人的 [BPW+12]（专注于讨论蒙特卡罗树搜索）；Mausam 和 Kolobov 的 [Kol12]（从人工智能的视角探讨马尔可夫决策问题）；Geffner 与 Bonet 的 [GB13]（研究搜索与自动规划问题）；Schmidhuber 的 [Sch15]、Arulkumaran 等所著的 [ADBB17]、Li 的 [Li17]、Busoniu 等著的 [BdBT+18]、Caterini 和 Chang 的 [CC18]（考虑了基于深度神经网络的强化学习方法）；Recht 的 [Rec19]（专注于讨论连续空间的控制问题）；还有笔者的 [Ber05]（专注于讨论策略前展算法与模型预测控制）、[Ber11]（专注于讨论近似策略迭代）和 [Ber18]（讨论了状态聚集的方法）、[Ber20a]（聚焦于多智能体问题）、[Ber24]（探讨强化学习与模型预测控制之间的关系）。

参考文献

[ADBB17] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[AF66] Michael Athans and Peter L. Falb. *Optimal Control*. McGraw-Hill, 1966.

[Ala22] Mazen Alamir. Learning against uncertainty in control engineering. *Annual Reviews in Control*, 53:19–29, 2022.

[BB23] Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer Nature, 2023.

[BBDSE17] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC press, 2017.

[BBM17] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[BBS91] Andrew Gehret Barto, Steven J. Bradtke, and Satinder P. Singh. Real-time learning and control using asynchronous dynamic programming. *Artificial Intelligence*, 72:82–138, 1991.

[BC99] Dimitri P. Bertsekas and David A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[BdBT+18] Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[Bel57] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Bel67] Richard E. Bellman. *Introduction to the Mathematical Theory of Control Processes*, volume I and II. Academic Press, 1967.

[Ber76] Dimitri P. Bertsekas. *Dynamic Programming and Stochastic Control*. Academic Press, 1976.

[Ber82] Dimitri P. Bertsekas. Distributed dynamic programming. *IEEE Transactions on Automatic Control*, 27(3):610–616, 1982.

[Ber83] Dimitri P. Bertsekas. Asynchronous distributed computation of fixed points. *Mathematical Programming*, 27(1):107–120, 1983.

[Ber91] Dimitri P. Bertsekas. *Linear Network Optimization: Algorithms and Codes*. MIT Press, 1991.

[Ber98] Dimitri P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[Ber05] Dimitri P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005.

[Ber11] Dimitri P. Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

[Ber12] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control: Vol. II*. Athena Scientific, 4th edition, 2012.

[Ber16] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena scientific, 3rd edition, 2016.

[Ber17] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control: Vol. I*. Athena scientific, 4th edition, 2017.

[Ber18] Dimitri P. Bertsekas. Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. *IEEE/CAA Journal of Automatica Sinica*, 6(1):1–31, 2018.

[Ber19a] Dimitri P. Bertsekas. Biased aggregation, rollout, and enhanced policy improvement for reinforcement learning. *arXiv preprint arXiv:1910.02426*, 2019.

[Ber19b] Dimitri P. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[Ber20a] Dimitri Bertsekas. Multiagent value iteration algorithms in dynamic programming and reinforcement learning. *Results in Control and Optimization*, 1:100003, 2020.

[Ber20b] Dimitri P. Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific, 2020.

[Ber21] Dimitri Bertsekas. Distributed asynchronous policy iteration for sequential zero-sum games and minimax control. *arXiv preprint arXiv:2107.10406*, 2021.

[Ber22a] Dimitri P. Bertsekas. *Abstract Dynamic Programming*. Athena Scientific, 3rd edition, 2022.

[Ber22b] Dimitri P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.

[Ber22c] Dimitri P. Bertsekas. Newton' s method for reinforcement learning and model predictive control. *Results in Control and Optimization*, 7:100121, 2022.

[Ber24] Dimitri P Bertsekas. Model predictive control and reinforcement learning: A unified framework based on dynamic programming. *IFAC-PapersOnLine*, 58(18):363–383, 2024.

[BH75] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Taylor & Francis, 1975.

[BI96] Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 14, 1996.

[Bit91] Sergio Bittanti. Count Riccati and the early days of the Riccati equation. In Sergio Bittanti, Alan J. Laub, and Jan C. Willems, editors, *The Riccati Equation*, pages 1–10. Springer, Berlin, 1991.

[BK65] Richard Ernest Bellman and Robert E. Kalaba. *Quasilinearization and Nonlinear Boundary-Value Problems*. Elsevier, 1965.

[Bla65] David Blackwell. Discounted dynamic programming. *The Annals of Mathematical Statistics*, 36(1):226–235, 1965.

[Bla67] David Blackwell. Positive dynamic programming. In *Proceedings of the 5th Berkeley symposium on Mathematical Statistics and Probability*, volume 1, pages 415–418. University of California Press Berkeley, 1967.

[BLW91] Sergio Bittanti, Alan J. Laub, and Jan C. Willems, editors. *The Riccati Equation*. Springer, Berlin, 1991.

[BMM25] Gokul Bhusal, Kevin Miller, and Ekaterina Merkurjev. MALADY: Multiclass active learning with auction dynamics on graphs. *IEEE Transactions on Artificial Intelligence*, 2025.

[Bor08] Vivek S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[Bor09] Vivek S. Borkar. Reinforcement learning—a bridge between numerical methods and Monte Carlo. In *Perspectives in Mathematical Sciences I: Probability and Statistics*, pages 71–91. World Scientific, 2009.

[BPW+12] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[Bra21] Paolo Brandimarte. *From Shortest Paths to Reinforcement Learning*. Springer, 2021.

[BS78] Dimitri P. Bertsekas and Steven Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978.

[BT89] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall Englewood Cliffs, NJ, 1989.

[BT96] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[BT08] Dimitri P. Bertsekas and John N Tsitsiklis. *Introduction to Probability*. Athena Scientific, 2nd edition, 2008.

[Cao07] Xi-Ren Cao. *Stochastic Learning and Optimization-A Sensitivity-Based Approach*. Springer, 2007.

[CC18] Anthony L. Caterini and Dong Eui Chang. *Deep Neural Networks in a Mathematical Framework*. Springer, 2018.

[CHFM13] Hyeong Soo Chang, Jiaqiao Hu, Michael C. Fu, and Steven I. Marcus. *Simulation-Based Algorithms for Markov Decision Processes*. Springer Science & Business Media, 2nd edition, 2013.

[DF04] Daniela Pucci De Farias. The linear programming approach to approximate dynamic programming. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Don Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[DNP13] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):388–403, 2013.

[FHS09] Franziska Feitzinger, Timo Hylla, and Ekkehard W. Sachs. Inexact Kleinman-Newton method for Riccati equations. *SIAM Journal on Matrix Analysis and Applications*, 31(2):272–288, 2009.

[FS04] Silvia Ferrari and Robert F. Stengel. Model-based adaptive critic designs. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Don Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[GB13] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[GBLB12] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[Gos15] Abhijit Gosavi. *Simulation-Based Optimization*. Springer, 2nd edition, 2015.

[Hay08] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education India, 3rd edition, 2008.

[Hew71] Gary Hewer. An iterative technique for the computation of the steady state gains for the discrete optimal regulator. *IEEE Transactions on Automatic Control*, 16(4):382–384, 1971.

[HR22] Moritz Hardt and Benjamin Recht. *Patterns, Predictions, and Actions: A Story About Machine Learning*. Princeton University Press, 2022.

[Hyl11] Timo Hylla. *Extension of inexact Kleinman-Newton methods to a general monotonicity preserving convergence theory*. PhD thesis, Dissertation, Trier, Universität Trier, 2011, 2011.

[JJ17] Yu Jiang and Zhong-Ping Jiang. *Robust Adaptive Dynamic Programming*. John Wiley & Sons, 2017.

[KAC+15] Mykel J. Kochenderfer, Christopher Amato, Girish Chowdhary, Jonathan P. How, Hayley J. Davison Reynolds, Jason R. Thornton, Pedro A. Torres-Carrasquillo, N. Kemal Üre, and John Vian. *Decision Making under Uncertainty: Theory and Application*. MIT press, 2015.

[KC16] Basil Kouvaritakis and Mark Cannon. *Model Predictive Control: Classical, Robust and Stochastic*. Springer, 2016.

[Kle68] David Kleinman. On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1):114–115, 1968.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kol12] Andrey Kolobov. Planning with Markov decision processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

[Kor90] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[Kri16] Vikram Krishnamurthy. *Partially Observed Markov Decision Processes*. Cambridge University Press, 2016.

[Li17] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[Lib11] Daniel Liberzon. *Calculus of Variations and Optimal Control theory: A Concise Introduction*. Princeton university press, 2011.

[LL13] Frank L. Lewis and Derong Liu. *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, volume 17. John Wiley & Sons, 2013.

[LLL08] Frank L. Lewis, Derong Liu, and George G. Lendaris. *Special Issue on Adaptive Dynamic Programming and Reinforcement Learning in Feedback Control*, volume 37 of *IEEE Transactions on Systems, Man, and Cybernetics, Part B*. IEEE, 2008.

[LR95] Peter Lancaster and Leiba Rodman. *Algebraic Riccati Equations*. Clarendon Press, 1995.

[LRD23] Cassidy Laidlaw, Stuart J. Russell, and Anca Dragan. Bridging RL theory and practice with the effective horizon. *Advances in Neural Information Processing Systems*, 36:58953–59007, 2023.

[LS20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[LV09] Frank L. Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[LWW+17] Derong Liu, Qinglai Wei, Ding Wang, Xiong Yang, and Hongliang Li. *Adaptive Dynamic Programming with Applications in Optimal Control*. Springer, 2017.

[MAMKF23] Franco Marchesoni-Acland, Jean-Michel Morel, Josselin Kherroubi, and Gabriele Facciolo. Optimal and efficient binary questioning for human-in-the-loop annotation. *arXiv preprint arXiv:2307.01578*, 2023.

[MDNMS01] Lalo Magni, Giuseppe De Nicolao, Lorenza Magnani, and Riccardo Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37(9):1351–1362, 2001.

[Mey08] Sean Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2008.

[Mey22] Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022.

[Pow11] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2nd edition, 2011.

[PR12] Warren B. Powell and Ilya O. Ryzhov. *Optimal Learning*. John Wiley & Sons, 2012.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[PVR04] Warren B. Powell and Benjamin Van Roy. Approximate dynamic programming for high-dimensional resource allocation problems. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Don Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[RDM+24] Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. Grandmaster-level chess without search. *CoRR*, 2024.

[Rec19] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

[RMD17] James B. Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing Madison, WI, 2nd edition, 2017.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[SBPW04] Jennie Si, Andrew G. Barto, Warren B. Powell, and Don Wunsch, editors. *Handbook of Learning and Approximate Dynamic Programming*, volume 2. IEEE Press, 2004.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[SGG+15] Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of Tetris. *Journal of Machine Learning Research*, 16:1629–1676, 2015.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[SHS+17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and Shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[SYL04] Jennie Si, Lei Yang, and Derong Liu. Direct neural dynamic programming. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Don Wunsch, editors, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[Sze10] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[Tes94] Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[Tes95] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[TG96] Gerald Tesauro and Gregory Galperin. On-line policy improvement using Monte Carlo search. *Advances in Neural Information Processing Systems*, 9:1068–1074, 1996.

[TVR96] John N. Tsitsiklis and Benjamin Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1):59–94, 1996.

[VEH20] Jesper E. Van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[VVL13] Draguna Vrabie, Kyriakos G. Vamvoudakis, and Frank L. Lewis. *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. IET, 2013.

[Wat89] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[WS92] David A. White and Donald A. Sofge, editors. *Handbook of Intelligent Control*. Van Nostrand Reinhold Company, 1992.

[YB15] Huizhen Yu and Dimitri P. Bertsekas. A mixed value and policy iteration method for stochastic control with universally measurable policies. *Mathematics of Operations Research*, 40(4):926–968, 2015.

[ZG09] Xiaojin Zhu and Andrew Goldberg. *Introduction to Semi-Supervised Learning.* Morgan & Claypool Publishers, 2009.

[Zha25] Shiyu Zhao. *Mathematical Foundations of Reinforcement Learning.* Springer, 2025.

[ZSGP20] Riccardo Zoppoli, Marcello Sanguineti, Giorgio Gnecco, and Thomas Parisini. *Neural Approximations for Optimal Control and Decision.* Springer, 2020.