强化学习知识点整理

Manus AI

2025 年 6 月 13 日

1 强化学习—-开学第一课—-余银峰—-v2

本文件主要介绍了强化学习的核心价值，包括目标设定与规划能力、试错与优化、探索未知与求真务实。强调了在强化学习过程中，目标的重要性、试错的必要性以及探索精神的价值。

核心价值

- **目标设定与规划能力**：明确的学习目标是成功的关键，它为学习提供方向和动力。规划能力帮助学生掌握设定短期与长期目标的方法，并认识到规划能力对个人成长和职业发展的重要性。这也能将目标和规划融入思政教育，培养学生的社会责任感和使命感。
- **试错与优化**：强化学习的过程并非一帆风顺，试错是不可避免的。从失败中汲取教训，不断改进策略，最终取得成功。面对失败，需要保持坚韧精神和积极向上的态度。
- **探索未知与求真务实**：勇于探索未知领域，关注前沿技术和发展趋势，并参与科研项目和实践活动，为未来的发展奠定坚实基础。保持求真务实的态度，认真分析问题，解决难题，挑战自己的极限。

成功要素

- **目标**：通往成功的灯塔，需要明确目标、制定详细计划并持之以恒。
- **挫折**：成功的垫脚石，应勇敢面对失败，不回避，不气馁，并总结经验，改进策略。

- **探索**: 创新的源泉，鼓励勇于尝试新的事物，挑战传统观念，在探索中不断创新，寻找最优策略，不断探索真理，追求卓越。

清晰的目标和合理的规划是成功的基础，勇于探索未知领域并不断创新，面对挫折永不放弃，坚持到底，是成为具有目标、坚韧不拔和探索精神的人的关键。

2 2-探索与利用

本文件深入探讨了强化学习中的核心概念：探索（Exploration）与利用（Exploitation），并详细介绍了多臂老虎机问题及其解决方案。

探索与利用

探索与利用是序列决策任务中的一个基本问题，旨在平衡获取已知最优收益与尝试不同决策之间的关系。

- **利用 (Exploitation)**: 执行能够获得已知最优收益的决策，即基于当前已知信息做出最佳选择。
- **探索 (Exploration)**: 尝试更多可能的决策，不一定会立即带来最优收益，但可能发现更好的策略。

策略探索原则

- **朴素方法 (Naive Exploration)**: 通过添加策略噪声，如 ϵ-greedy 策略，以一定的概率进行随机探索。
- **积极初始化 (Optimistic Initialization)**: 给动作的初始价值一个较高的估计，鼓励智能体在早期进行更多探索。
- **基于不确定性的度量 (Uncertainty Measurement)**: 尝试具有不确定收益的策略，例如使用上置信界 (UCB) 算法，该算法倾向于选择那些被探索次数较少或具有较高潜在价值的动作。
- **概率匹配 (Probability Matching)**: 基于概率选择最佳策略，如 Thompson Sampling，根据每个动作成为最优的概率来选择动作。

多臂老虎机问题 (Multi-arm Bandit)

多臂老虎机问题是研究探索与利用技术理论的最佳环境，可以被视为无状态（state-less）强化学习。其目标是最大化累积收益。

收益估计

期望收益 $Q_n(a)$ 和采样次数 $N(a)$ 的关系：

$$Q_n(a) = \frac{\sum_{i=1}^{n} R_i(a)}{N(a)}$$

其中 $R_i(a)$ 是第 i 次选择动作 a 获得的收益。

Regret 函数

Regret 函数衡量了决策与最优决策之间的收益差。总 Regret 函数 $O_T = E[\sum_{t=1}^{T} R(a_t)]$。理论上，渐近最优 Regret 为 $O(\log T)$。

常用算法

- **贪心策略 (Greedy Strategy)**: 总是选择当前估计收益最高的动作，但可能陷入局部最优。
- **ϵ-greedy 策略**: 以 $1-\epsilon$ 的概率选择当前最优动作，以 ϵ 的概率随机选择动作。ϵ 可以随时间衰减（decaying ϵ-greedy），理论上可以达到对数渐近收敛。
- **UCB (Upper Confidence Bounds) 算法**: 策略选择 $a = \arg\max_a [Q(a) + U(a)]$，其中 $U(a)$ 是不确定性项，鼓励探索不确定性大的动作。UCB 算法在理论上具有收敛性。
- **Thompson Sampling 方法**: 根据每个动作成为最优的概率来选择动作，通过从动作价值的后验分布中采样来选择动作。

总结

探索与利用是强化学习中试错（trial-and-error）过程的必备技术。多臂老虎机是研究这些技术的理想环境，并且 ϵ-greedy、UCB 和 Thompson

Sampling 方法在多臂老虎机任务和强化学习的探索中都非常常用，其中 ϵ-greedy 最为常见。

3 3-马尔可夫决策过程

本文件详细介绍了马尔可夫决策过程(Markov Decision Process, MDP)，它是强化学习中用于建模决策过程的数学框架，以及基于动态规划求解 MDP 的方法。

随机过程与马尔可夫过程

- **随机过程**: 描述一个或多个事件、随机系统或随机现象随时间演变的过程。概率论研究静态统计规律，而随机过程研究动态统计规律。
- **马尔可夫过程 (Markov Process)**: 具有马尔可夫性质的随机过程。马尔可夫性质指"未来独立于过去，只依赖于当前状态"(The future is independent of the past given the present)。这意味着当前状态包含了所有与未来预测相关的充分信息。

马尔可夫决策过程 (MDP)

MDP 形式化地描述了一种强化学习环境，其特点是环境完全可观测，且当前状态完全表征过程（马尔可夫性质）。

MDP 五元组

一个 MDP 可以由一个五元组 (S, A, P, γ, R) 表示：

- S: 状态的集合，例如迷宫中的位置、Atari 游戏中的屏幕显示。
- A: 动作的集合，例如移动方向、手柄操作。
- P: 状态转移概率，对每个状态 $s \in S$ 和动作 $a \in A$，$P(s'|s, a)$ 是下一个状态 s' 在 S 中的概率分布。
- $\gamma \in [0, 1]$: 对未来奖励的折扣因子，使得智能体更重视即时奖励。
- $R : S \times A \to \mathbb{R}$: 奖励函数，有时奖励只与状态相关，表示为 $R(s)$。

MDP 的动态

MDP 的动态过程从初始状态 s_0 开始，智能体选择动作 a_0，获得奖励 $R(s_0, a_0)$，然后 MDP 随机转移到下一个状态 s_1，如此循环，直到终止状态或无限进行。累积奖励为 $R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \dots$。

占用度量 (Occupancy Measure)

占用度量 $p^\pi(s, a)$ 表示在策略 π 下，状态-动作对 (s, a) 被访问的频率。它与策略密切相关，不同的策略会产生不同的状态-行动对分布。策略的累积奖励可以通过占用度量和奖励函数的乘积求和得到。

基于动态规划的强化学习

当 MDP 模型已知时（即状态转移 P 和奖励函数 R 明确给出），可以通过动态规划的方法计算最优价值函数和学习最优策略。

价值函数与 Bellman 等式

- **价值函数** $V^\pi(s)$: 表示从状态 s 开始，并遵循策略 π 所能获得的预期累积奖励。
- **Bellman 等式**: 描述了价值函数与其后续状态价值之间的关系：

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V^\pi(s')$$

- **最优价值函数** $V^*(s)$: 在所有策略中，状态 s 所能获得的最大可能折扣奖励之和。
- **最优策略** $\pi^*(s)$: 能够最大化累积奖励期望的策略。

价值迭代与策略迭代

- **价值迭代 (Value Iteration)**: 通过迭代更新价值函数直到收敛来找到最优价值函数。它是一种贪心更新法，在计算中没有明确的策略。
- **策略迭代 (Policy Iteration)**: 包含两个步骤：策略评估（估计当前策略的价值函数）和策略改进（根据价值函数更新策略）。对于状态空间较小的 MDP，策略迭代通常收敛较快。

基于模型的强化学习

在实际问题中，状态转移和奖励函数通常不是明确给出的。基于模型的强化学习旨在从经验数据中学习 MDP 模型（估计状态转移概率 P 和奖励函数 R），然后利用这些估计的模型执行动态规划来求解最优策略。另一种方法是模型无关的强化学习，直接从经验中学习价值函数和策略，而不显式学习 MDP 模型。

总结

MDP 是强化学习的核心数学框架，由五元组 (S, A, P, γ, R) 构成。当环境已知时，可使用价值迭代和策略迭代等动态规划方法求解最优策略。当环境未知时，可以通过统计信息拟合 MDP 模型，或直接采用模型无关的强化学习方法。

4 4-值函数估计

本文件主要介绍了无模型强化学习中的值函数估计方法，包括蒙特卡洛方法和时序差分学习，并对两者进行了比较。

无模型强化学习 (Model-free RL)

在现实问题中，通常无法明确给出状态转移和奖励函数。无模型强化学习直接从经验中学习值（value）和策略（policy），而无需构建马尔可夫决策过程模型（MDP）。其关键步骤包括估计值函数和优化策略。

值函数估计

在模型无关的强化学习中，我们无法直接获得状态转移概率 P 和奖励函数 R，但可以通过一系列经验来估计值函数。

蒙特卡洛方法 (Monte-Carlo Methods)

- **原理**: 依赖于重复随机抽样来获得数值结果，例如通过模拟围棋对弈来估计当前状态下的胜率。

- **价值估计**: 从策略 π 下的经验片段中学习 $V^\pi(s)$。值函数被估计为累计奖励（return）的均值。累计奖励 $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$。
- **实现**: 使用策略 π 采样片段，增量更新计数器 $N(s)$ 和总累计奖励 $S(s)$，最终 $V(s) = S(s)/N(s)$。当 $N(s) \to \infty$ 时，$V(s) \to V^\pi(s)$。
- **特点**: 直接从经验片段进行学习，是模型无关的，从完整的片段中学习（不使用 bootstrapping），只能应用于有限长度的 MDP。
- **增量更新**: 对于每个状态 s_t 和对应累计奖励 G_t，更新公式为 $V(s_t) \leftarrow V(s_t) + \alpha(G_t - V(s_t))$。
- **离线策略蒙特卡洛**: 使用重要性采样（importance sampling）来评估一个不同于生成经验的策略。通过重要性比率对累计奖励进行加权。

时序差分学习 (Temporal Difference Learning, TD)

- **原理**: 通过下一步的价值估计来更新当前一步的价值估计。TD 目标为 $R_{t+1} + \gamma V(s_{t+1})$，TD 误差为 $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$。
- **更新**: $V(s_t) \leftarrow V(s_t) + \alpha(R_{t+1} + \gamma V(s_{t+1}) - V(s_t))$。
- **特点**: 直接从经验片段中学习，是模型无关的，通过 bootstrapping 从不完整的片段中学习，可以进行在线学习，能够从不完整的序列中学习，适用于连续（无终止的）环境。
- **离线策略时序差分**: 使用重要性采样对时序差分目标进行加权，仅需要一步来进行重要性采样修正，具有比蒙特卡洛重要性采样更低的方差。

蒙特卡洛 (MC) 与时序差分 (TD) 的比较

两者都旨在从策略 π 下的经验片段学习 $V^\pi(s)$，但存在显著差异:

- **学习方式**: MC 必须等待片段结束才能更新，TD 可以在每一步之后进行在线学习。
- **完整性**: MC 只能从完整序列中学习，TD 可以从不完整的序列中学习。

- **偏差与方差**:
  - MC 具有高方差、无偏差的特点，收敛性质良好，对初始值不敏感，易于理解和使用。
  - TD 具有低方差、有偏差的特点，通常比 MC 更高效，但对初始值更敏感。
- **反向传播 (Backup)**: MC 是“深层”反向传播，TD 是“浅层”反向传播。动态规划是全宽度反向传播。

多步时序差分学习

多步时序差分学习结合了蒙特卡洛和时序差分的优点，定义了 n 步累计奖励 $G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(s_{t+n})$，并进行相应的更新。

总结

无模型强化学习在黑盒环境下使用，其核心任务是精准高效地估计状态或状态-动作对的价值。蒙特卡洛方法通过采样到底的方式直接估计价值函数，而时序差分学习通过下一步的价值估计来更新当前一步的价值估计。在实际使用中，时序差分方法更为常见。

5 5-Sarsa

本文件主要介绍了强化学习中的两种重要的无模型控制算法：Sarsa 和 Q-learning，并对它们进行了详细的对比。

从价值函数到行动控制

在强化学习中，仅仅知道什么是好的（即估计值函数 $V^\pi(s)$）是不够的，还需要知道如何做好行动。当环境模型未知时，直接估计状态-动作值函数 $Q^\pi(s, a)$ 对于直接进行行动控制（选择最佳动作）至关重要。因为 $Q^\pi(s, a)$ 直接关联了在特定状态下采取特定动作的预期累积奖励，从而可以直接用于策略改进。

Sarsa 算法

Sarsa 是一种在线策略（on-policy）时序差分控制算法。这意味着它使用当前策略来选择动作并进行学习。

Sarsa 更新规则

Sarsa 的名称来源于其更新规则所依赖的五元组：$(S, A, R, Sprime, Aprime)$，即“状态-动作-奖励-新状态-新动作”。其状态-动作值函数的更新公式为：

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(Sprime, Aprime) - Q(S, A)]$$

其中：

- S, A: 当前状态和当前采取的动作。
- R: 采取动作 A 后获得的奖励。
- $Sprime, Aprime$: 转移到的下一个状态和在该新状态下根据当前策略选择的下一个动作。
- α: 学习率。
- γ: 折扣因子。

Sarsa 控制算法

Sarsa 算法在每个时间步长进行策略评估和策略改进：

- **策略评估**: 使用 Sarsa 更新规则更新 $Q(S, A)$。
- **策略改进**: 通常采用 ϵ-greedy 策略，从当前 Q 值中派生出新的行为策略。

Q-learning 算法

Q-learning 是一种离线策略（off-policy）学习方法。这意味着它的学习过程可以独立于行为策略，即它学习的是最优策略的价值函数，而不管智能体实际遵循的是什么策略。

Q-learning 更新规则

Q-learning 的更新目标是下一个状态-动作对的最大 Q 值，而不是下一个状态下实际采取动作的 Q 值。其状态-动作值函数的更新公式为：

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_{Aprime} Q(Sprime, Aprime) - Q(S,A)]$$

其中：

- S, A: 当前状态和当前采取的动作。
- R: 采取动作 A 后获得的奖励。
- $Sprime$: 转移到的下一个状态。
- $\max_{Aprime} Q(Sprime, Aprime)$: 下一个状态 $Sprime$ 下所有可能动作中最大的 Q 值。
- α: 学习率。
- γ: 折扣因子。

Q-learning 控制算法

Q-learning 的目标策略是关于 $Q(s,a)$ 的贪心策略，而行为策略通常是 ϵ-greedy 策略。Q-learning 能够收敛到最优状态-动作值函数 $Q^*(s,a)$。

Sarsa 与 Q-learning 的对比

- **策略类型**: Sarsa 是在线策略（on-policy），Q-learning 是离线策略（off-policy）。
- **更新目标**: Sarsa 使用 $Q(Sprime, Aprime)$ 作为更新目标，其中 $Aprime$ 是根据当前策略选择的动作。Q-learning 使用 $\max_{Aprime} Q(Sprime, Aprime)$ 作为更新目标，即选择下一个状态下的最优动作。
- **安全性**: 在“悬崖漫步”（Cliff-walking）等环境中，Sarsa 倾向于学习更安全的路径，因为它考虑了实际采取的动作。Q-learning 可能会学习到一条“最优”但危险的路径，因为它总是假设会采取最优动作。

- **收敛性**: Q-learning 能够收敛到最优策略，即使行为策略是探索性的。Sarsa 只有在行为策略逐渐收敛到贪心策略时才能收敛到最优策略。

总结

Sarsa 和 Q-learning 都是强化学习中重要的无模型控制算法。Sarsa 是一种在线策略算法，学习当前策略的价值；Q-learning 是一种离线策略算法，学习最优策略的价值。在实际应用中，选择哪种算法取决于具体任务的需求，例如对安全性的考量。

6 6-多步 Sarsa

本文件深入探讨了多步时序差分（TD）学习方法，以及如何将其应用于 Sarsa 算法，并介绍了使用重要性采样的离线多步学习方法和多步树回溯算法。

多步时序差分预测

回顾了动态规划（DP）和时序差分（TD）的关系，强调了 TD 学习是采样反向传播，而 DP 是完全反向传播。多步 TD 预测结合了蒙特卡洛方法和单步 TD 学习的优点，通过考虑多步的未来奖励来更新价值函数。

n 步累计奖励

定义 n 步累计奖励 $G_t^{(n)}$ 为:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

其中，$V(S_{t+n})$ 是对未来状态价值的估计。n 步时序差分学习的更新公式为:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t^{(n)} - V(S_t)]$$

当 $n = 1$ 时，退化为单步 TD；当 $n \to \infty$ 时，近似于蒙特卡洛方法。

平均 n 步累计奖励 (TD(λ))

TD(λ) 算法通过对所有不同步长的累计奖励进行加权平均来更新价值函数，权重随步长呈指数衰减。λ 参数控制了蒙特卡洛和单步 TD 之间的平衡。当 $\lambda = 1$ 时，相当于蒙特卡洛方法；当 $\lambda = 0$ 时，相当于单步 TD 方法。

多步 Sarsa

将多步 TD 的思想应用于 Sarsa 算法，得到了 n 步 Sarsa 算法。其核心思想是使用 n 步累计奖励来更新状态-动作值函数 $Q(S, A)$：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[G_t^{(n)} - Q(S_t, A_t)]$$

其中，$G_t^{(n)}$ 是基于 n 步奖励和 $Q(S_{t+n}, A_{t+n})$ 估计的 n 步累计奖励。

使用重要性采样的多步离线学习

在离线学习场景中，行为策略（用于收集数据）和目标策略（用于优化）可能不同。为了修正这种差异，引入了重要性采样。对于多步离线学习，重要性采样系数 $\rho_{t:t+n-1}$ 用于调整 n 步累计奖励的权重，以反映目标策略下轨迹的概率与行为策略下轨迹的概率之比。

多步树回溯算法 (N-step Tree Backup Algorithm)

多步树回溯算法是一种不需要使用重要性采样的离线算法，它结合了 Q-learning 和期望 Sarsa 的思想。该算法通过对未来状态的期望 Q 值进行回溯来更新当前状态-动作值函数，从而避免了重要性采样带来的方差问题。

总结

多步时序差分方法通过结合多步信息，在偏差和方差之间取得了更好的平衡。多步 Sarsa 将这一思想应用于控制问题。而多步离线学习方法，如使用重要性采样的算法和多步树回溯算法，则解决了在行为策略和目标策略不同时的学习问题，为强化学习提供了更灵活和高效的解决方案。

7 7-Temporal Difference Learning

本文件主要介绍了时序差分（Temporal Difference, TD）学习，包括其预测方法和控制方法，并与蒙特卡洛（MC）方法进行了比较。

TD 预测

TD 预测旨在针对给定策略 π 计算状态值函数 $V^{\pi}(s)$。与蒙特卡洛方法不同，TD 学习不需要等到整个片段结束才进行更新，而是利用下一个状态的估计值来更新当前状态的值。

TD 方法的特点

- **Bootstrapping**: TD 学习使用估计值来更新估计值，即“自举”。蒙特卡洛方法不自举，而动态规划和 TD 都自举。
- **Sampling**: TD 学习通过采样而不是期望值来更新，与蒙特卡洛方法类似。动态规划不采样。
- **模型无关**: TD 方法不需要环境模型，仅需要经验。
- **增量学习**: TD 方法可以完全增量式学习，即在知道最终结果之前就可以学习，所需内存和计算峰值更少。
- **不完整序列学习**: TD 方法可以在没有最终结果的情况下学习，适用于不完整序列和连续（无终止）环境。

TD 与 MC 的比较

- **收敛性**: 在批量更新（Batch Updating）下，TD(0) 对于有限马尔可夫预测任务会收敛，但与蒙特卡洛方法收敛到的答案可能不同。
- **偏差与方差**: TD 学习通常具有比蒙特卡洛方法更低的方差，但可能存在偏差。
- **实际应用**: 在随机游走等示例中，TD 学习通常表现出更快的收敛速度和更好的性能。

TD 控制

TD 学习不仅可以用于预测（策略评估），还可以扩展到控制（策略优化）。

Sarsa: 在线策略 TD 控制

Sarsa 是一种在线策略 TD 控制算法，它使用当前策略来选择动作并更新状态-动作值函数 $Q(s, a)$。其更新规则为：

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(Sprime, Aprime) - Q(S, A)]$$

其中 $Aprime$ 是在 $Sprime$ 状态下根据当前策略选择的动作。

Q-Learning: 离线策略 TD 控制

Q-Learning 是一种离线策略 TD 控制算法，它学习的是最优策略的价值函数，而不管智能体实际遵循的是什么策略。其更新规则为：

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_{Aprime} Q(Sprime, Aprime) - Q(S, A)]$$

Q-Learning 的目标是下一个状态下所有可能动作中的最大 Q 值。

Sarsa 与 Q-Learning 的对比

在“悬崖漫步”（Cliffwalking）等任务中，Sarsa 倾向于学习更安全的路径，因为它考虑了实际采取的动作。Q-Learning 可能会学习到一条“最优”但危险的路径，因为它总是假设会采取最优动作。

Actor-Critic 方法

Actor-Critic 方法结合了策略（Actor）和价值函数（Critic）的显式表示。Critic 学习价值函数，Actor 根据 Critic 的反馈更新策略。这种方法具有最小化动作选择计算、可以学习显式随机策略以及可以对策略施加约束等优点。

总结

时序差分学习是强化学习中的核心概念，它提供了一种高效的从经验中学习价值函数和优化策略的方法。TD 学习的自举和采样特性使其在许多实际应用中优于蒙特卡洛方法。Sarsa 和 Q-Learning 是两种重要的 TD 控制算法，分别代表了在线策略和离线策略学习的范例。Actor-Critic 方法则进一步结合了策略和价值函数的显式表示，为复杂任务提供了更灵活的解决方案。

8 9-规划与学习 (1)

本文件主要探讨了强化学习中的规划与学习，包括模型、规划的定义，以及如何将规划和学习集成到 Dyna 算法中。

策略评估与策略提升

策略评估 (Policy Evaluation)

给定环境 MDP 和策略 π，策略值函数 $V^\pi(s)$ 和状态-动作值函数 $Q^\pi(s,a)$ 可以通过 Bellman 方程进行估计。它们分别表示从状态 s 或状态-动作对 (s,a) 开始，并遵循策略 π 所能获得的预期累积奖励。

策略提升 (Policy Improvement)

策略提升是指找到一个新策略 π'，使得对于任何状态 s，新策略的价值函数 $V^{\pi'}(s)$ 大于或等于原策略的价值函数 $V^\pi(s)$。策略提升定理指出，如果 $Q^\pi(s,\pi'(s)) \geq V^\pi(s)$ 对于所有状态 s 成立，那么 $V^{\pi'}(s) \geq V^\pi(s)$ 也成立，即新策略 π' 优于或等于原策略 π。

模型 (Model)

模型能够预测给定状态和动作的下一个状态和奖励的分布。模型分为:

- **分布模型 (Distribution Model)**: 描述了轨迹的所有可能性及其概率。
- **样本模型 (Sample Model)**: 根据概率进行采样，只产生一条可能的轨迹。

规划 (Planning)

规划是一个输入模型，输出策略的搜索过程。规划的通用框架是通过模型采样得到模拟数据，然后利用模拟数据更新值函数从而改进策略。动态规划是一种规划方法，它搜索整个状态空间，生成所有的状态转移分布，并回溯更新状态的值函数。

规划与学习 (Planning and Learning)

- **规划**: 利用模型产生的模拟经验。
- **学习**: 利用环境产生的真实经验。

两者都通过回溯（back-up）更新值函数的估计。学习的方法也可以用在模拟经验上。

Dyna 算法

Dyna 算法集成了规划、决策和学习，旨在通过结合直接强化学习和模型学习来提高效率。

Dyna 架构

Dyna 架构包括三个主要部分：

- **直接强化学习**: 使用真实经验更新值函数和策略。
- **模型学习**: 使用真实经验更新环境模型。
- **基于模型的规划**: 基于模型随机采样得到模拟经验，并使用模拟经验进行规划，更新值函数和策略。

Dyna-Q 算法

Dyna-Q 算法是一种表格型 Dyna 算法，它在每个时间步执行以下步骤：

1. 智能体在环境中执行动作，获得真实奖励和下一个状态，并进行一次直接强化学习更新（例如 Q-learning）。
2. 模型学习模块使用真实经验更新环境模型。

3. 规划模块重复 n 次：随机选择一个之前见过的状态和动作，使用模型预测下一个状态和奖励，然后进行一次基于模型的 Q-learning 更新。

Dyna 的优势与挑战

- **优势**: 减少对真实经验的需求，加速学习过程。
- **挑战**: 当环境是随机的、模型泛化性不好或环境发生变化时，Dyna 算法可能面临挑战。例如，在“阻碍迷宫”和“捷径迷宫”的例子中，Dyna-Q+（加入了探索机制）能够更好地适应环境变化并发现最优路径。

总结

规划与学习是强化学习中重要的研究方向。Dyna 算法通过集成直接强化学习、模型学习和基于模型的规划，有效地平衡了探索与利用，提高了学习效率。然而，在复杂和动态的环境中，如何构建准确的模型并有效利用规划仍然是需要深入研究的问题。

9 10-规划与学习 (2)

本文件继续深入探讨了强化学习中的规划与学习，重点介绍了更高效的采样方法、实时动态规划以及决策时规划，并最终引出了蒙特卡洛树搜索。

高效采样方法

优先级采样 (Prioritized Sweeping)

- **问题**: 均匀随机采样效率低下，因为很多模拟经验和更新可能集中在不重要的状态-动作对上。
- **思想**: 根据值函数改变的幅度定义优先级，优先更新那些变化大的状态-动作对。通过设置优先级更新队列，可以加速收敛。
- **优势**: 收敛更快，尤其在确定性环境中表现显著。

期望更新 (Expected Updates) 与采样更新 (Sample Updates)

- **期望更新**: 需要分布模型，计算量大，但没有偏差，更准确。适用于分支因子较小的情况。
- **采样更新**: 只需要采样模型，计算量小，但存在采样误差。适用于大的随机分支因子和状态数量较多的情况。

轨迹采样 (Trajectory Sampling)

(Content truncated due to size limit. Use line ranges to read in chunks)

- **思想**: 在状态空间中按照特定分布采样，根据当前策略下所观测的分布进行采样，而不是遍历整个状态空间。
- **优势**: 不需要知道当前策略下状态的分布，计算量少，简单有效。
- **缺点**: 可能不断重复更新已经被访问的状态。

实时动态规划 (Real-time Dynamic Programming, RTDP)

- **区别于传统动态规划**: RTDP 采用实时的轨迹采样，只更新轨迹访问的状态值。
- **优势**: 能够跳过策略无关的状态，在解决状态集合规模大的问题上具有优势，满足一定条件下可以以概率 1 收敛到最优策略。
- **应用**: 在“跑道问题”等任务中，RTDP 相比传统动态规划，在收敛到最优策略所需的更新次数上表现出显著优势。

决策时规划 (Decision-time Planning)

- **背景规划 (Background Planning)**: 规划是为了更新很多状态值供后续动作的选择（如动态规划、Dyna）。
- **决策时规划**: 规划只着眼于当前状态的动作选择。在不需要快速反应的应用中很有效，如棋类游戏。

启发式搜索 (Heuristic Search)

- **思想**: 访问当前状态（根节点），对后续可能的情况进行树结构展开，叶节点代表估计的值函数，然后回溯到当前状态。
- **特点**: 决策时规划，着重于当前状态。搜索越深，计算量越大，得到的动作越接近最优。性能提升源于专注当前状态的后续可能。

Rollout 算法

- **思想**: 从当前状态进行模拟的蒙特卡洛估计，选取最高估计值的动作，并在下一个状态重复上述步骤。
- **目的**: 类似于策略迭代和改进，寻找更优的策略。
- **加速方法**: 可以通过多个处理器并行采样、轨迹截断（用存储的值估计代替回报）以及排除不可能成为最佳动作的动作来加速。

蒙特卡洛树搜索 (Monte Carlo Tree Search, MCTS)

MCTS 是一种结合了蒙特卡洛控制和决策时规划的算法，广泛应用于棋类游戏（如 AlphaGo）。其主要步骤包括：

1. **选择 (Selection)**: 根据树策略（动作值函数）遍历树到一个叶节点。
2. **扩展 (Expansion)**: 从选择的叶节点出发，选择未探索过的动作到达新的状态。
3. **模拟 (Simulation)**: 从新的状态出发，按照 rollout 策略进行轨迹模拟。
4. **回溯 (Backup)**: 得到的回报回溯更新树策略，rollout 访问的状态值不会被保存。

重复上述步骤直至计算资源耗尽，从根节点选择最优动作。MCTS 保留了过去一部分的经验数据，下一个状态树的初始树是上一个状态树具有高回报的部分。

总结

规划与学习是强化学习中的重要组成部分。本文件介绍了多种高效的采样方法，如优先级采样、期望更新与采样更新、轨迹采样。同时，探讨了实时动态规划和决策时规划，以及启发式搜索、Rollout 算法和蒙特卡洛树搜索等具体算法。这些方法在不同场景下提供了更高效、更灵活的强化学习解决方案，尤其在复杂决策问题中展现出强大潜力。

10 11-近似逼近方法 (1)

本文件主要介绍了强化学习中处理大规模或连续状态/动作空间的近似逼近方法，包括状态/动作的离散化与分桶，以及参数化值函数近似。

大规模 MDP 的挑战

在处理大规模马尔可夫决策过程(MDP)时,传统的基于查询表(Lookup Table）的方法（如维护 $V(s)$ 或 $Q(s,a)$）变得不可行，因为状态或状态-动作空间可能非常大，甚至是连续的（例如围棋博弈、直升机控制)。

解决方案

状态/动作的离散化与分桶

- **离散化**: 对于连续状态 MDP，可以将状态空间划分为网格，转化为离散状态值。例如，二维连续状态 (s_1, s_2) 可以通过网格切分转化为离散状态。
- **分桶**: 对于大型离散状态 MDP，可以进一步对状态值进行分桶或采样聚合，利用先验知识将相似的离散状态归类到一起。
- **优点**: 操作简洁直观，高效，在许多问题中能达到较好效果。
- **缺点**: 可能过于简单地表示价值函数，假设每个离散区间为常数值，且面临“维度灾难”问题。

参数化值函数近似

- **思想**: 构建参数化（可学习的）函数来近似值函数，例如 $V_\theta(s) \approx V^\pi(s)$ 或 $Q_\theta(s,a) \approx Q^\pi(s,a)$，其中 θ 是近似函数的参数。
- **优势**: 能够将现有可见的状态泛化到没有见过的状态上，适用于非稳态、非独立同分布的数据训练。
- **主要形式**: 可以是线性模型、神经网络、决策树、最近邻、傅立叶/小波基底等。其中，线性模型和神经网络更适合强化学习的非稳态数据特性。

基于随机梯度下降（SGD）的值函数近似

目标是找到参数向量 θ 以最小化值函数近似值与真实值之间的均方误差 $J(\theta) = \mathbb{E}[(V^\pi(s) - V_\theta(s))^2]$。通过随机梯度下降，参数 θ 沿着误差减小的梯度方向更新:

$$\theta \leftarrow \theta + \alpha[V^\pi(s) - V_\theta(s)]\nabla_\theta V_\theta(s)$$

其中 α 是学习率，$[V^\pi(s) - V_\theta(s)]$ 是预测误差，$\nabla_\theta V_\theta(s)$ 是特征值。

线性状态值函数近似

使用特征的线性组合表示价值函数 $V_\theta(s) = \theta^T x(s)$，其中 $x(s)$ 是状态 s 的特征向量。目标函数是参数 θ 的二次函数，因此随机梯度下降能够收敛到全局最优解。

蒙特卡洛状态值函数近似

对于每个数据样本 (s_t, G_t)，参数更新为:

$$\theta \leftarrow \theta + \alpha[G_t - V_\theta(s_t)]\nabla_\theta V_\theta(s_t)$$

蒙特卡洛预测至少能收敛到一个局部最优解，在线性价值函数的情况下可以收敛到全局最优。

时序差分状态值函数近似

对于每个数据样本 $(s_t, r_{t+1} + \gamma V_\theta(s_{t+1}))$，参数更新为：

$$\theta \leftarrow \theta + \alpha[r_{t+1} + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)]\nabla_\theta V_\theta(s_t)$$

线性情况下时序差分学习（接近）收敛到全局最优解。

状态-动作值函数近似

对动作-状态值函数进行近似 $Q_\theta(s,a) = \theta^T x(s,a)$，其中 $x(s,a)$ 是状态-动作对的特征向量。更新方式与状态值函数近似类似，例如对于 Sarsa 算法，目标是 $r_{t+1} + \gamma Q_\theta(s_{t+1}, a_{t+1})$。

总结

近似逼近方法是强化学习处理大规模和连续状态/动作空间的关键。通过离散化、分桶和参数化值函数近似，可以将强化学习算法应用于更复杂的实际问题。基于随机梯度下降的线性近似方法在理论上具有良好的收敛性，为深度强化学习奠定了基础。

11 12-近似逼近方法 (2)

本文件继续深入探讨了强化学习中的近似逼近方法，重点介绍了深度 Q 网络（DQN）、策略梯度方法及其改进，以及 Actor-Critic 算法。

案例分析: Deep Q-Network (DQN)

DQN 是一种将深度学习与 Q-learning 相结合的方法，用于解决大规模状态空间下的强化学习问题。其核心思想是使用深度神经网络来近似 Q 函数 $Q(s,a;\theta)$。

DQN 的主要改进

- **经验回放 (Experience Replay)**: 将智能体与环境交互产生的经验 (s,a,r,s') 存储在经验池 D 中，然后从中随机均匀采样小批量数据进行训练。这打破了数据之间的时序相关性，提高了数据利用效率，并避免了在近期经验上过拟合。

- **目标网络 (Target Network)**: 使用两个 Q 网络：一个是在线 Q 网络（参数为 θ），用于计算当前 Q 值；另一个是目标 Q 网络（参数为 θ^-），用于计算目标 Q 值。目标网络的参数 θ^- 会定期（例如每 C 步）从在线 Q 网络复制而来，而不是每一步都更新。这使得训练过程更加稳定。

DQN 的损失函数通常定义为：

$$L(\theta) = \mathbb{E}(s, a, r, s') \sim U(D)[(r + \gamma \max Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

其中 $r + \gamma \max_{a'} Q(s', a'; \theta^-)$ 是目标 Q 值，而 $Q(s, a; \theta)$ 是预测 Q 值。

策略梯度 (Policy Gradient)

策略梯度方法直接学习参数化的策略 $\pi_\theta(a|s)$，而不是学习值函数。策略可以是确定性的，也可以是随机的。

基于策略的强化学习的特点

- **收敛性质更好**: 通常具有更好的收敛性质。
- **适用于高维/连续动作空间**: 在高维度或连续动作空间中更有效，因为基于值函数的方法通常需要取最大值操作，在高维空间中计算复杂。
- **能够学习随机策略**: 可以学习出随机策略，这在某些部分可观测或需要探索的环境中非常有用。
- **缺点**: 通常会收敛到局部最优而非全局最优；评估一个策略通常不够高效且具有较大的方差。

策略梯度定理

策略梯度定理将似然比的推导过程泛化到多步马尔可夫决策过程。对于任意可微的策略 $\pi_\theta(a|s)$，任意策略的目标函数 $J(\theta)$ 的策略梯度为：

$$\nabla_\theta J(\theta) = \mathbb{E}\pi\theta[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$$

其中 $Q^{\pi_\theta}(s, a)$ 是在策略 π_θ 下状态-动作对 (s, a) 的真实值函数。

蒙特卡洛策略梯度 (REINFORCE)

REINFORCE 算法是一种蒙特卡洛策略梯度方法，它利用随机梯度上升更新策略参数。它使用累计奖励值 G_t 作为 $Q^{\pi_\theta}(s,a)$ 的无偏采样。

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log \pi_\theta(a_t|s_t)$$

REINFORCE 存在的问题

- **基于片段式数据**: 通常需要任务有终止状态才能直接计算累计折扣奖励。
- **低数据利用效率**: 需要大量的训练数据。
- **高训练方差**: 从单个或多个片段中采样到的值函数具有很高的方差，这是其最重要的缺陷。

Softmax 随机策略

Softmax 策略是一种常用的随机策略，其概率定义为:

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

其中 $f_\theta(s,a)$ 是用 θ 参数化的状态-动作对得分函数。其对数似然的梯度可以推导得到。

Actor-Critic

Actor-Critic 方法结合了策略梯度和值函数学习。它包含两个核心组件:

- **演员 (Actor)**: 参数化策略 $\pi_\theta(a|s)$，负责选择动作。
- **评论家 (Critic)**: 参数化值函数 $Q_w(s,a)$ 或 $V_w(s)$，负责评估演员策略所采取动作的价值。

评论家学习准确估计当前演员策略的动作价值，而演员则根据评论家的反馈（例如 $Q_w(s,a)$）通过策略梯度更新其策略参数。

A2C: Advantage Actor-Critic

A2C 算法通过引入优势函数 (Advantage Function) 来标准化评论家的打分，进一步降低训练方差:

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

策略梯度更新变为:

$$\nabla_\theta J(\theta) = \mathbb{E}\pi\theta[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)]$$

由于 $Q^{\pi_\theta}(s,a) = \mathbb{E}[R+\gamma V^{\pi_\theta}(s')|s,a]$,因此优势函数也可以表示为 $A^{\pi_\theta}(s,a) = R + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$。这意味着我们只需要拟合状态值函数 $V_w(s)$ 就可以估计优势函数。

总结

价值和策略的近似逼近方法是强化学习从理论走向实际应用的关键。参数化的价值函数和策略使得强化学习能够处理大规模和连续的问题。通过链式法则，价值函数的参数可以直接学习。通过似然比方法，可以使用优势函数来学习策略的参数。Actor-Critic 框架同时学习了价值函数和策略，通过价值函数的 Q 值（或优势）估计，以策略梯度的方式更新策略参数，有效结合了两类方法的优点。

12 13-深度强化学习价值方法

本文件主要介绍了深度强化学习中基于价值的方法，特别是深度 Q 网络（DQN）及其重要的改进版本：Double DQN 和 Dueling DQN。

深度强化学习简介

深度强化学习结合了深度学习的感知能力和强化学习的决策能力，使得强化学习算法能够以端到端的方式解决复杂问题，例如直接从原始像素输入中学习玩 Atari 游戏。然而，深度强化学习也面临挑战，如高维参数空间、训练稳定性、数据需求量大以及计算资源平衡等问题。深度强化学习方法主要分为:

- **基于价值的方法**: 如深度 Q 网络（DQN）及其扩展。

- **基于随机策略的方法**: 如策略梯度、TRPO、PPO 等。
- **基于确定性策略的方法**: 如 DPG、DDPG 等。

深度 Q 网络 (DQN)

DQN 使用神经网络来近似 Q 函数 $Q(s,a;\theta)$。然而，直接将神经网络应用于 Q learning 会导致训练不稳定，主要原因包括：

- **连续采样**: 连续的经验数据 $(s_t, a_t, r_{t+1}, s_{t+1})$ 之间存在时序相关性，不满足神经网络训练所需的独立同分布假设。
- **目标 Q 值不稳定**: 目标 Q 值 $r+\gamma \max_{aprime} Q(sprime, aprime; \theta)$ 依赖于正在更新的 Q 网络参数 θ，导致目标值频繁变动，使得训练不稳定。

为解决这些问题，DQN 引入了两个关键技术：

经验回放 (Experience Replay)

将智能体与环境交互产生的经验 $(s, a, r, sprime)$ 存储在经验池 D 中。训练时，从经验池中随机均匀采样小批量数据进行训练。这打破了数据之间的时序相关性，提高了数据利用效率，并避免了在近期经验上过拟合。优先经验回放（Prioritized Experience Replay）进一步优化了采样过程，根据 TD 误差的大小来衡量经验的重要性，优先采样那些学习价值更大的经验。

目标网络 (Target Network)

使用两个 Q 网络：一个是在线 Q 网络（参数为 θ），用于计算当前 Q 值；另一个是目标 Q 网络（参数为 θ^-），用于计算目标 Q 值。目标网络的参数 θ^- 会定期（例如每 C 步）从在线 Q 网络复制而来，而不是每一步都更新。这使得目标值保持相对稳定，从而稳定了训练过程。DQN 的损失函数为：

$$L(\theta) = \mathbb{E}(s, a, r, sprime) \sim U(D)[(r+\gamma \max Q(sprime, aprime; \theta^-) - Q(s, a; \theta))^2]$$

Double DQN

传统的 Q-learning 存在 Q 函数过高估计的问题。这是因为在计算目标 Q 值时，max 操作会倾向于选择被高估的动作值，导致 Q 值不断向上累积，偏离真实值。数学上，$\mathbb{E}[\max(X_1, X_2)] \geq \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$。Double DQN 旨在解决 Q 函数的过高估计问题。它通过解耦动作选择和动作评估来消除这种偏差：

- **动作选择**：使用在线 Q 网络选择下一个状态 $sprime$ 的最优动作 $aprime = \arg\max_{aprime} Q(sprime, aprime; \theta)$。
- **动作评估**：使用目标 Q 网络评估所选动作的 Q 值 $Q(sprime, aprime; \theta^-)$。

Double DQN 的目标 Q 值计算公式为：

$$Y_t^{DoubleDQN} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_{aprime} Q(s_{t+1}, aprime; \theta); \theta^-)$$

实验表明，Double DQN 能够有效降低 Q 值的过高估计，从而提高算法的性能和稳定性。

Dueling DQN

Dueling DQN 是一种改进的 DQN 架构，它将 Q 函数分解为状态价值函数 $V(s)$ 和优势函数 $A(s, a)$：

$$Q(s, a) = V(s) + A(s, a)$$

其中 $V(s)$ 表示状态 s 的价值，而 $A(s, a)$ 表示在状态 s 下采取动作 a 相对于平均水平的优势。为了解决 $V(s)$ 和 $A(s, a)$ 的不可辨识性问题（即 $V(s)$ 和 $A(s, a)$ 的组合可以产生相同的 $Q(s, a)$），通常会进行归一化处理，例如：

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \alpha) + (A(s, a; \theta, \beta) - \frac{1}{|\mathcal{A}|} \sum_{aprime} A(s, aprime; \theta, \beta))$$

其中 θ 是共享网络参数，α 和 β 分别是价值流和优势流的独有参数。Dueling DQN 的优势在于：

- **处理与动作关联较小的状态**：即使在动作不影响环境的状态下，Dueling DQN 也能有效学习状态的价值，因为价值流 $V(s)$ 不依赖于具体的动作。

- **状态值函数的学习更有效**：一个状态值函数对应多个优势函数，使得网络能够更好地学习状态的通用特征，从而提高学习效率。
- **提高探索效率**：在某些探索任务中，Dueling DQN 能够更有效地评估状态的重要性，从而指导智能体进行更有效的探索。

总结

深度强化学习中的价值方法，特别是 DQN 及其变体，为解决大规模和复杂环境下的强化学习问题提供了强大的工具。经验回放和目标网络解决了 DQN 的训练稳定性问题，Double DQN 解决了 Q 值的过高估计问题，而 Dueling DQN 则通过分解 Q 函数提高了学习效率和探索能力。这些方法的不断发展推动了深度强化学习在游戏、机器人控制等领域的广泛应用。

13 14-深度强化学习策略方法 (1)

本文件主要介绍了深度强化学习中基于策略的方法，包括基于神经网络的策略梯度、异步优势 Actor-Critic (A3C) 方法以及确定性策略梯度 (DPG) 及其深度版本 DDPG。

深度强化学习策略方法概述

深度强化学习方法可以分为基于价值的方法（如 DQN 及其扩展）、基于随机策略的方法（如策略梯度、TRPO、PPO、A3C）和基于确定性策略的方法（如 DPG、DDPG）。本文件主要关注后两者。

基于神经网络的策略梯度

策略梯度方法直接学习参数化的策略 $\pi_\theta(a|s)$。策略网络的梯度可以通过链式法则和似然比方法推导得到。对于随机策略，通常使用 Softmax 函数实现动作概率，其参数化得分函数可以通过神经网络实现。

策略梯度与 Q 学习的对比

- **Q 学习**：学习一个 Q 函数 $Q_\theta(s,a)$，优化目标是最小化 TD error。更新方程基于 Bellman 方程。

- **策略梯度**: 直接学习策略 $\pi_\theta(a|s)$，优化目标是最大化策略的价值。更新方程基于策略梯度定理，通过对 $\log \pi_\theta(a|s)$ 乘以优势函数或 Q 值进行更新。

A3C: 异步优势 Actor-Critic

A3C（Asynchronous Advantage Actor-Critic）是一种异步的 Actor-Critic 方法，它结合了策略梯度和值函数学习，并通过异步并行训练来提高效率和稳定性。

A3C 的特点

- **异步 (Asynchronous)**: 多个并行的智能体（worker）在各自的环境副本中独立地收集经验并计算梯度，然后异步地更新全局网络的参数。这打破了数据之间的相关性，提高了训练效率。
- **优势 (Advantage)**: 策略梯度的更新使用优势函数 $A(s,a) = Q(s,a) - V(s)$，这有助于降低方差并提供更准确的梯度估计。
- **Actor Critic**: 包含一个 Actor（策略网络）和一个 Critic（值函数网络），两者协同工作。Critic 评估 Actor 策略的价值，Actor 根据 Critic 的反馈更新策略。

A3C 在 Atari 游戏等任务中表现出色，并且由于其异步特性，可以在 CPU 上高效运行，减少了对 GPU 的依赖。

确定性策略梯度 (Deterministic Policy Gradient, DPG)

DPG 是一种用于连续动作空间的策略梯度方法，它直接学习一个确定性策略 $\mu_\theta(s)$，而不是随机策略。DPG 的策略梯度定理为:

$$\nabla_\theta J(\theta) = \mathbb{E}s \sim \rho^\mu[\nabla\theta\mu_\theta(s)\nabla_a Q^\mu(s,a)|a = \mu\theta(s)]$$

其中 ρ^μ 是策略 μ 下的状态访问分布，$\nabla_a Q^\mu(s,a)$ 是 Q 函数对动作的梯度。

深度确定性策略梯度 (Deep Deterministic Policy Gradient, DDPG)

DDPG 是 DPG 的深度学习版本，它将深度神经网络应用于 DPG，使其能够处理高维状态和动作空间。DDPG 借鉴了 DQN 的成功经验，引入了以下关键技术来提高训练的稳定性和性能：

- **经验回放 (Experience Replay)**：存储经验并从中随机采样，打破数据相关性。
- **目标网络 (Target Network)**：使用独立的 Actor 目标网络和 Critic 目标网络，定期更新，以稳定训练目标。
- **批标准化 (Batch Normalization)**：在 Q 网络中应用批标准化，有助于稳定训练。
- **探索噪声**：在 Actor 的输出中添加噪声，以确保足够的探索。

DDPG 算法流程：

1. 随机初始化 Critic 网络 $Q(s, a|\theta^Q)$ 和 Actor 网络 $\mu(s|\theta^\mu)$。
2. 初始化目标网络 Q^{target} 和 μ^{target} 的权重，并初始化经验回放缓冲区。
3. 对于每个 episode：
   - 初始化探索噪声。
   - 对于每个时间步：
     - 根据当前策略和探索噪声选择动作 $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$。
     - 执行动作，观察奖励 r_t 和新状态 s_{t+1}。
     - 将经验 (s_t, a_t, r_t, s_{t+1}) 存储到经验回放缓冲区。
     - 从缓冲区中随机采样一个 minibatch 的经验。
     - 计算目标 Q 值：

$$y_i = r_i + \gamma Q^{\text{target}}(s_{i+1}, \mu^{\text{target}}(s_{i+1}|\theta^{\mu^{\text{target}}})|\theta^{Q^{\text{target}}})$$

     - 更新 Critic 网络，最小化损失函数：

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

– 更新 Actor 网络：

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}$$

– 软更新目标网络：

$$\theta^{Q^{\text{target}}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q^{\text{target}}}$$
$$\theta^{\mu^{\text{target}}} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu^{\text{target}}}$$

实验表明，目标网络对于 DDPG 的性能至关重要。

总结

深度强化学习中的策略方法，特别是基于神经网络的策略梯度、A3C 和 DDPG，为解决复杂连续控制任务提供了强大的工具。A3C 通过异步并行训练提高了效率和稳定性，而 DDPG 则将确定性策略梯度与深度学习技术相结合，使其能够处理高维连续动作空间。这些方法的不断发展推动了强化学习在机器人控制、自动驾驶等领域的应用。

14 15-深度强化学习策略方法 (2)

本文件主要介绍了深度强化学习中基于策略的方法，特别是信任区域策略优化（TRPO）和近端策略优化（PPO），它们旨在解决传统策略梯度方法中步长难以确定和训练不稳定的问题。

策略梯度的缺点

传统的策略梯度方法（如 REINFORCE）存在以下缺点：

- **步长难以确定**：策略的微小变化可能导致性能的巨大波动，选择合适的学习率非常困难。
- **数据分布变化**：策略的更新会导致采样数据的分布发生变化，使得训练不稳定。

信任区域策略优化 (TRPO)

TRPO 旨在通过限制策略更新的幅度来保证策略性能的单调提升。其核心思想是在每次策略更新时，限制新策略与旧策略之间的 KL 散度在一个小的信任区域内。这可以表示为一个带约束的优化问题：

$$\begin{aligned} \max_{\theta'} \quad & \mathbb{E}_{s\sim\rho^{\pi},a\sim\pi}\left[\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}A^{\pi_\theta}(s,a)\right] \\ \text{s.t.} \quad & \mathbb{E}_{s\sim\rho^{\pi}}\left[D_{\mathrm{KL}}(\pi_\theta(\cdot|s)\|\pi_{\theta'}(\cdot|s))\right] \le \delta \end{aligned}$$

其中 $A^{\pi_\theta}(s,a)$ 是优势函数，δ 是 KL 散度的最大允许值。TRPO 通过共轭梯度法等复杂优化技术来求解这个约束优化问题。

近端策略优化 (PPO)

PPO 是一种在 TRPO 基础上进行改进的策略优化算法，它在保持 TRPO 性能优势的同时，大大简化了算法的实现和计算复杂度。PPO 的核心思想是使用一个裁剪函数限制重要性采样比率。

裁剪式目标函数 (Clipped Objective)

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$

其中 $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ 是重要性采样比率，A_t 是优势函数，ϵ 是裁剪范围。

自适应 KL 惩罚项 (Adaptive KL Penalty)

$$L^{\mathrm{KLPEN}}(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}A_t - \beta D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}(\cdot|s_t)\|\pi_\theta(\cdot|s_t))\right]$$

其中 β 是 KL 惩罚项的系数。

PPO 相较于 TRPO 的改进

- **实现简单**: 使用一阶优化替代复杂的共轭梯度法。
- **计算高效**: 无需估算 Fisher 信息矩阵。
- **性能优越**: 在多个基准环境中优于 TRPO。

PPO 通常结合 GAE（广义优势估计）提高稳定性，并使用 mini-batch SGD 并行更新策略。

总结

TRPO 和 PPO 是深度策略梯度优化的重要成果。TRPO 在理论上具有严格的收敛性保证，而 PPO 在实践中以其简洁、高效和稳定性成为当前最受欢迎的策略优化算法之一。这些方法推动了强化学习在实际复杂环境中的成功应用。