《强化学习：原理和 Python 实现》目录
Reinforcement Learning: Theory and Python Implementation

3.1.2. 压缩映射和 Bellman 算子	3.1.2 Contraction Mapping and Bellman Operator
3.1.3. Banach 不动点定理	3.1.3 Banach Fix-point Theorem
3.2. 有模型策略迭代	3.2 Model-based Policy Iteration
3.2.1. 策略评估	3.2.1 Policy Evaluation
3.2.2. 策略改进	3.2.2 Policy Improvement
3.2.3. 策略迭代	3.2.3 Policy Iteration
3.3. 有模型价值迭代	3.3 Model-based Value Iteration
3.4. 动态规划	3.4 Dynamic Programming (DP)
3.4.1. 从动态规划看迭代算法	3.4.1 Interpret Iteration Algorithm using DP
3.4.2. 异步动态规划	3.4.2 Asynchronous DP
3.5. 案例：冰面滑行	3.5 Case Study: FrozenLake
3.5.1. 实验环境的使用	3.5.1 Usage of Environment
3.5.2. 有模型策略迭代求解	3.5.2 Use Policy Iteration to Solve
3.5.3. 有模型价值迭代求解	3.5.3 Use Value Iteration to Solve
3.6. 本章小结	3.6 Summary
4. 回合更新价值迭代	4 Monte Carlo Learning
4.1. 同策回合更新	4.1 On-policy Monte Carlo Learning
4.1.1. 同策回合更新策略评估	4.1.1 On-policy Monte Carlo Policy Evaluation
4.1.2. 带起始探索的同策回合更新	4.1.2 Monte Carlo Learning with Exploration Start
4.1.3. 基于柔性策略的同策回合更新	4.1.3 Monte Carlo Learning on Soft Policy
4.2. 异策回合更新	4.2 Off-policy Monte Carlo Learning
4.2.1. 重要性采样	4.2.1 Importance Sampling
4.2.2. 异策回合更新策略评估	4.2.2 Off-policy Monte Carlo Policy Evaluation
4.2.3. 异策回合更新最优策略评估	4.2.3 Off-policy Monte Carlo Policy Optimization
4.3. 实验：21 点游戏	4.3 Case Study: Blackjack
4.3.1. 游戏环境使用	4.3.1 Usage of Environment
4.3.2. 同策策略评估	4.3.2 On-policy Policy Evaluation
4.3.3. 同策最优策略求解	4.3.3 On-policy Policy Optimization
4.3.4. 异策策略评估	4.3.4 Off-policy Policy Evaluation
4.3.5. 异策最优策略求解	4.3.5 Off-policy Policy Optimization
4.4. 本章小结	4.4 Summary
5. 时序差分价值迭代	5 Temporal Difference (TD) Learning
5.1. 同策时序差分更新	5.1 On-policy TD Learning
5.1.1. 时序差分更新策略评估	5.1.1 TD Policy Evaluation
5.1.2. SARSA 算法	5.1.2 SARSA

5.1.3. 期望 SARSA 算法	5.1.3 Expected SARSA
5.2. 异策时序差分控制	5.2 Off-policy TD Learning
5.2.1. 基于重要性采样的异策算法	5.2.1 Off-policy Algorithm based on Importance Sampling
5.2.2. Q 学习	5.2.2 Q Learning
5.2.3. 双重 Q 学习	5.2.3 Double Q Learning
5.3. 资格迹	5.3 Eligibility Trace
5.3.1. λ 回报	5.3.1 λ Return
5.3.2. TD(λ)	5.3.2 TD(λ)
5.4. 案例：的士调度	5.4 Case Study: Taxi
5.4.1. 实验环境使用	5.4.1 Usage of Environment
5.4.2. 同策时序差分学习调度	5.4.2 Use On-policy TD to Solve
5.4.3. 异策时序差分学习调度	5.4.3 Use Off-policy TD to Solve
5.4.4. 资格迹学习调度	5.4.4 Use Eligibility Trace to Solve
5.5. 本章小结	5.5 Summary
6. 函数近似方法	6 Function Approximation
6.1. 函数近似原理	6.1 Basic of Function Approximation
6.1.1. 随机梯度下降	6.1.1 Stochastic Gradient Descent
6.1.2. 半梯度下降	6.1.2 Semi-Stochastic Gradient Descent
6.1.3. 带资格迹的半梯度下降	6.1.3 Semi-Stochastic Gradient Descent with Eligibility Trace
6.2. 线性近似	6.2 Linear Approximation
6.2.1. 精确查找表与线性近似的关系	6.2.1 Compare Lookup Table and Linear Approximation
6.2.2. 线性最小二乘策略评估	6.2.2 Least Linear Squared Policy Evaluation
6.2.3. 线性最小二乘最优策略求解	6.2.3 Least Linear Squared Policy Optimization
6.3. 函数近似的收敛性	6.3 Convergence of Function Approximation
6.4. 深度 Q 学习	6.4 Deep Q Learning
6.4.1. 经验回放	6.4.1 Experience Replay
6.4.2. 带目标网络的深度 Q 学习	6.4.2 Deep Q Learning with Target Network
6.4.3. 双重深度 Q 网络	6.4.3 Double Q Network
6.4.4. 对偶深度 Q 网络	6.4.4 Dual Q Network
6.5. 案例：小车上山	6.5 Case Study: MountainCar
6.5.1. 实验环境使用	6.5.1 Use the Environment
6.5.2. 用线性近似求解最优策略	6.5.2 Use Function Approximation to Solve
6.5.3. 用深度 Q 学习求解最优策略	6.5.3 Use Deep Q Learning to Solve the Task
6.6. 本章小结	6.6 Summary

7. 回合更新策略梯度方法	7 Policy Gradient (PG)
7.1. 策略梯度算法的原理	7.1 Theory of PG
7.1.1. 函数近似与动作偏好	7.1.1 Function Approximation and Action Preference
7.1.2. 策略梯度定理	7.1.2 PG Theorem
7.2. 同策回合更新策略梯度算法	7.2 On-policy PG Algorithm
7.2.1. 简单的策略梯度算法	7.2.1 Vanilla PG
7.2.2. 带基线的简单策略梯度算法	7.2.2 PG with Baseline
7.3. 异策回合更新策略梯度算法	7.3 Off-policy PG Algorithm
7.4. 策略梯度更新和极大似然估计的关系	7.4 Relationship between PG and Maximum Likelihood
7.5. 案例：车杆平衡	7.5 Case Study: CartPole
7.5.1. 同策策略梯度算法求解最优策略	7.5.1 Use On-policy PG to Control
7.5.2. 策策略梯度算法求解最优策略	7.5.2 Use Off-policy PG to Control
7.6. 本章小结	7.6 Summary
8. 执行者/评论者方法	8 Actor-Critic (AC)
8.1. 同策执行者/评论者算法	8.1 On-policy AC Algorithms
8.1.1. 动作价值执行者/评论者算法	8.1.1 Action-Value AC
8.1.2. 优势执行者/评论者算法	8.1.2 Advantage AC
8.1.3. 带资格迹的执行者/评论者算法	8.1.3 Eligibility Trace AC
8.2. 基于代理优势的同策方法	8.2 On-policy Algorithm with Surrogate Objective
8.2.1. 代理优势	8.2.1 Surrogate Objective
8.2.2. 邻近策略优化	8.2.2 Proximal Policy Optimization
8.3. 信任域算法	8.3 Trust Region Algorithm
8.3.1. KL 散度	8.3.1 Kullback-Leibler Divergence
8.3.2. 信任域	8.3.2 Trust Region Method
8.3.3. 自然策略梯度算法	8.3.3 Natural PG
8.3.4. 信任域策略优化	8.3.4 Trust Region Policy Optimization
8.3.5. Kronecker 因子信任域执行者/评论者算法	8.3.5 AC using Kronecker-factored Trust Region
8.4. 重要性采样异策执行者/评论者算法	8.4 Importance Sampling Off-policy AC Algorithms
8.4.1. 基本的异策算法	8.4.1 Off-policy Vanilla PG
8.4.2. 带经验回放的异策算法	8.4.2 Off-policy PG with Experience Replay
8.5. 柔性执行者/评论者算法	8.5 Soft AC Algorithm
8.5.1. 熵	8.5.1 Entropy
8.5.2. 奖励工程和带熵的收益	8.5.2 Reward Engineering and Energy-based Reward
8.5.3. 柔性执行者/评论者算法的网络设计	8.5.3 Network Design of Soft AC
8.6. 案例：双节倒立摆	8.6 Case Study: Acrobot
8.6.1. 同策执行者/评论者算法求解最优策	8.6.1 Use On-policy AC to Solve

略	
8.6.2. 异策执行者/评论者算法求解最优策略	8.6.2 Use Off-policy AC to Solve
8.7. 本章小结	8.7 Summary
9. 连续动作空间的确定性策略	9 Deterministic Policy Gradient (DPG)
9.1. 同策确定性算法	9.1 On-policy DPG
9.1.1. 策略梯度定理的确定性版本	9.1.1 DPG Theorem
9.1.2. 基本的同策确定性执行者/评论者算法	9.1.2 DPG Algorithm
9.2. 异策确定性算法	9.2 Off-policy DPG
9.2.1. 基本的异策确定性执行者/评论者算法	9.2.1 Vanilla Off-policy DPG
9.2.2. 深度确定性策略梯度算法	9.2.2 Deep DPG
9.2.3. 双重延迟深度确定性策略梯度算法	9.2.3 Twin Delay Deep DPG
9.3. 案例：倒立摆的控制	9.3 Case Study: Pendulum
9.3.1. 用深度确定性策略梯度算法求解	9.3.1 Use Deep DPG to Solve
9.3.2. 用双重延迟深度确定性算法求解	9.3.2 Use Twin Delay Deep DPG to Solve
9.4. 本章小结	9.4 Summary
10. 综合案例：电动游戏	10 Case Study: Video Game
10.1. Atari 游戏环境	10.1 Atari Game Environment
10.1.1. Gym 库的完整安装	10.1.1 Complete Installation of Gym
10.1.2. 游戏环境使用	10.1.2 Usage of Game Environment
10.2. 基于深度 Q 学习的游戏 AI	10.2 Play Games using Deep Q Learning
10.2.1. 算法设计	10.2.1 Design of Implementation
10.2.2. 智能体的实现	10.2.2 Implementation of Agent
10.2.3. 智能体的训练和测试	10.2.3 Train and Test the Agent
10.3. 本章小结	10.3 Summary
11. 综合案例：棋盘游戏	11 Case Study: Board Game
11.1. 双人确定性棋盘游戏	11.1 Deterministic 2-Player Board Game
11.1.1. 五子棋和井字棋	11.1.1 Gobang and Tic Tac Toe
11.1.2. 黑白棋	11.1.2 Reversi
11.1.3. 围棋	11.1.3 Game of Go
11.2. AlphaZero 算法	11.2 AlphaZero Algorithm
11.2.1. 回合更新树搜索	11.2.1 Monte Carlo Tree Search
11.2.2. 深度残差网络	11.2.2 Deep Residual Network
11.2.3. 自我对弈	11.2.3 Self-Play
11.2.4. 算法流程	11.2.4 Put Everything Together
11.3. 棋盘游戏环境 boardgame2	11.3 Board Game Environment

	boardgame2
11.3.1. 为 Gym 库扩展自定义环境	11.3.1 Extend Gym for Customed Environment
11.3.2. boardgame2 设计	11.3.2 Design of boardgame2
11.3.3. Gym 环境接口的实现	11.3.3 Implement APIs for Gym
11.3.4. 树搜索接口的实现	11.3.4 Implement APIs for Tree Search
11.4. AlphaZero 算法的实现	11.4 Implement AlphaZero
11.4.1. 智能体类的实现	11.4.1 Implement Agent
11.4.2. 算法流程的实现	11.4.2 Implement Workflow
11.5. 本章小结	11.5 Summary
12. 综合案例：自动驾驶	12 Case Study: Autonomous Driving
12.1. AirSim 开发环境使用	12.1 Introduce AirSim
12.1.1. 安装和运行	12.1.1 Install and Run AirSim
12.1.2. 用 Python 访问 AirSim	12.1.2 AirSim's Python APIs
12.2. 基于强化学习的自动驾驶	12.2 RL Driver
12.2.1. 为自动驾驶设计强化学习环境	12.2.1 Convert Autonomous Driving to an RL Task
12.2.2. 智能体的设计和实现	12.2.2 Design and Implement Agent
12.2.3. 智能体的训练和测试	12.2.3 Train and Test Agent
12.3. 本章小结	12.3 Summary