强化学习及其应用

Reinforcement Learning and Its Applications

第二章 动态规划方法

Dynamic Programming Methods

授课人：周晓飞
zhouxiaofei@iie.ac.cn
2023-6-12

第二章 动态规划方法

2.1 Bellman 公式

2.2 动态规划：策略收敛法

2.3 动态规划：值迭代法

2.4 值迭代方法比较

第二章 动态规划方法

2.1 Bellman 公式

2.2 动态规划：策略收敛法

2.3 动态规划：值迭代法

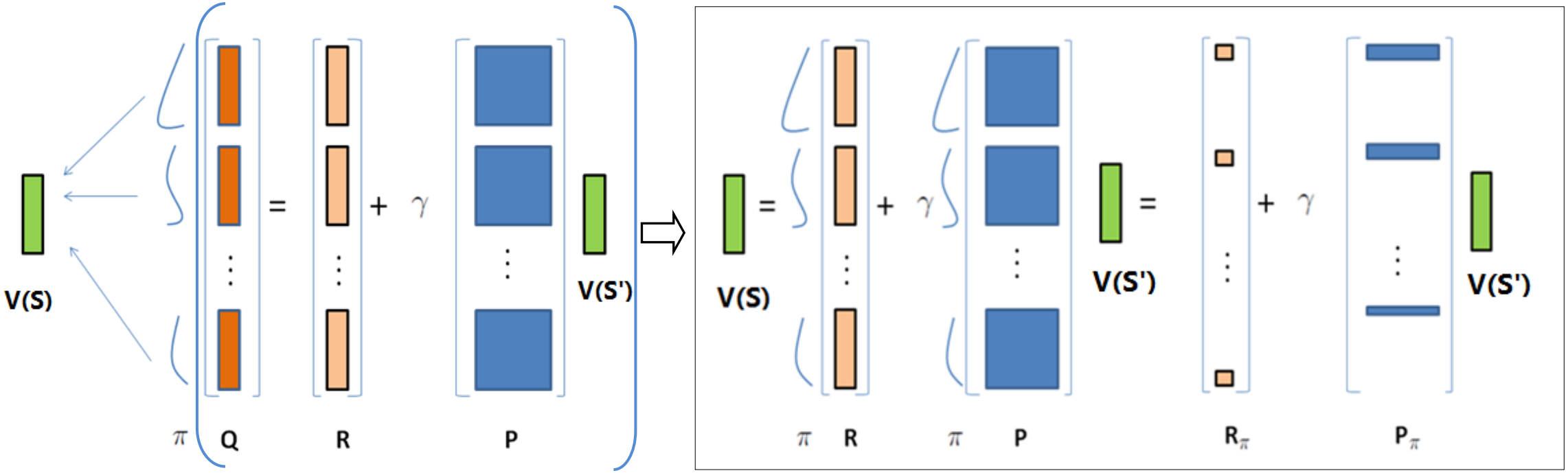
2.4 值迭代方法比较

Bellman 公式

Bellman Expectation Equation for V

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

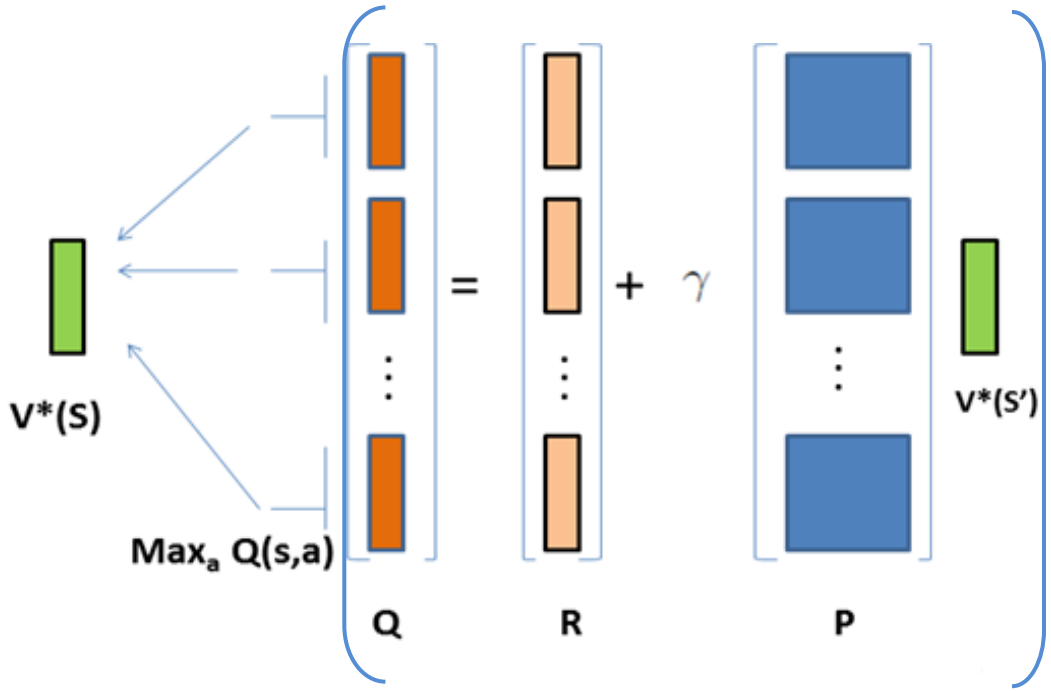
$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$



Bellman 公式

Bellman Optimality Equation for v_*

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$



Bellman 公式

动态规划

- A method for solving complex problems
- By breaking them down into subproblems
  - Solve the subproblems
  - Combine solutions to subproblems

DP 方法: 已知 Bellman 方程的环境参数(回报 R 和转移概率 P),

求取最优策略 u*和最优值 v*。

第二章 动态规划方法

2.1 Bellman 公式

2.2 动态规划：策略收敛法

2.3 动态规划：值迭代法

2.4 值迭代方法比较

DP策略收敛法

算法

初始化最优策略 u，

Step1: 确定的最优策略u，以V = Vin=Vout，求解V；

Step2: V→Vin，得到Q因子

Step3: 对 S_i 选择最优a， $\max_a Q(S_i, a)$，形成最优策略u；

Step4: goto Step1，直到最优策略u不变。

D P策略收敛法

算法

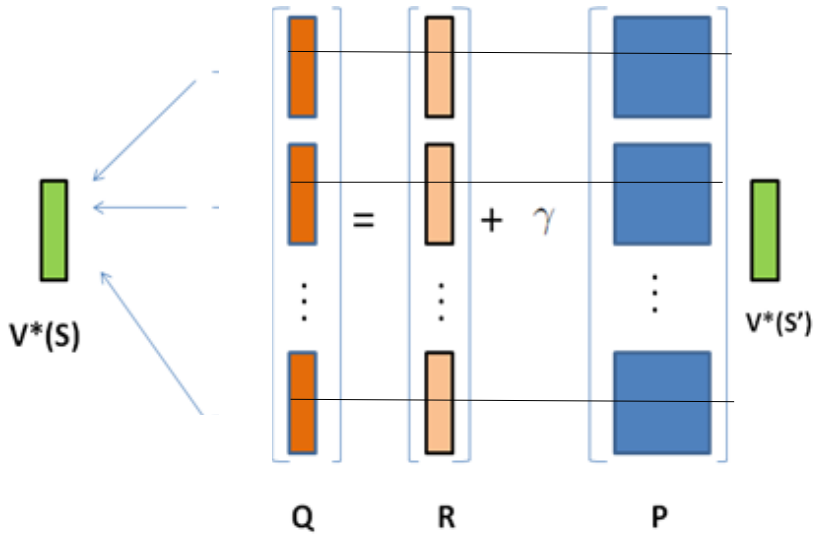
初始化最优策略 u，

Step1：确定的最优策略u，以V = V_{in}=V_{out}，求解V；

Step2：V→Vin，得到Q因子

Step3：对 S_i 选择最优 a， $max_a Q(S_i, a)$，形成最优策略 u；

Step4: goto Step1，直到最优策略 u 不变。



D P策略收敛法

算法

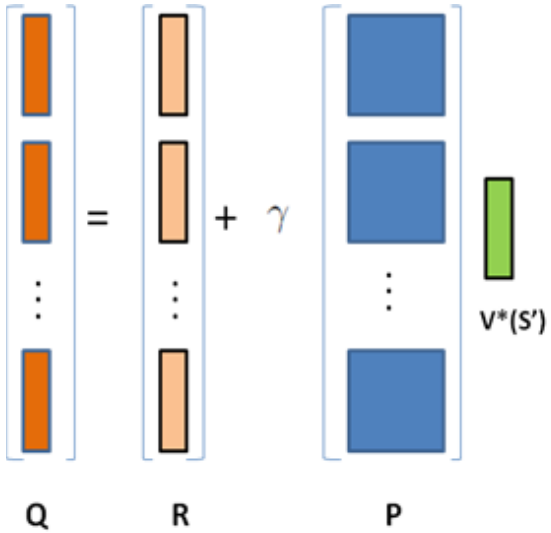
初始化最优策略 u，

Step1：确定的最优策略 u，以 V = V_{in}=V_{out}，求解 V；

Step2：V→Vin，得到Q因子

Step3：对 S_i 选择最优 a，$max_a Q(S_i, a)$，形成最优策略 u；

Step4: goto Step1，直到最优策略 u 不变。



D P策略收敛法

算法

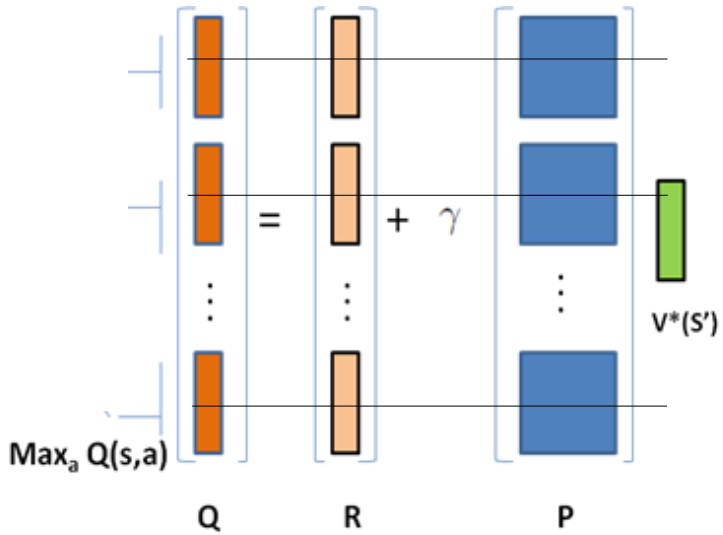
初始化最优策略 u，

Step1：确定的最优策略 u，以 V = V_{in}=V_{out}，求解 V；

Step2：V→Vin，得到Q因子

Step3：对 S_i 选择最优 a，$max_a Q(S_i, a)$，形成最优策略 u；

Step4: goto Step1，直到最优策略 u 不变。



D P策略收敛法

算法

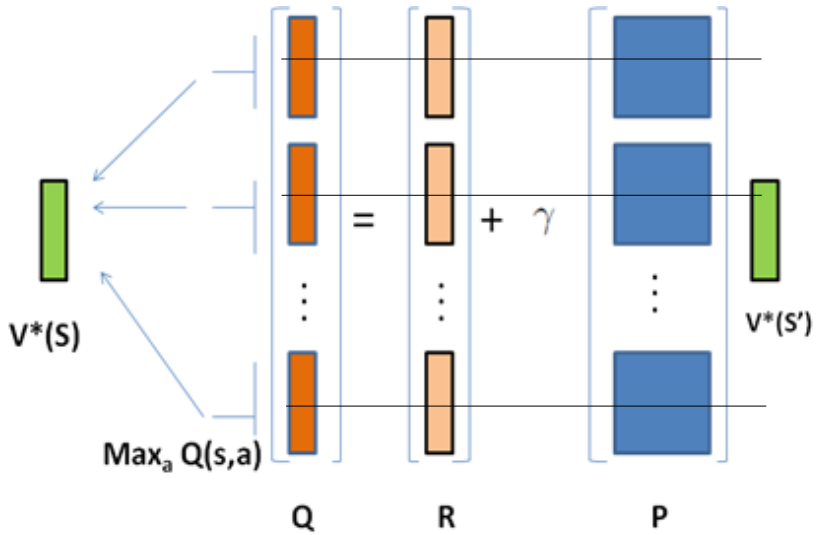
初始化最优策略 u，

Step1：确定的最优策略 u，以 V = V_{in}=V_{out}，求解 V；

Step2：V→Vin，得到Q因子

Step3：对 S_i 选择最优 a，$max_a Q(S_i, a)$，形成最优策略 u；

Step4: goto Step1，直到最优策略 u 不变



第二章 动态规划方法

2.1 Bellman 公式

2.2 动态规划：策略收敛法

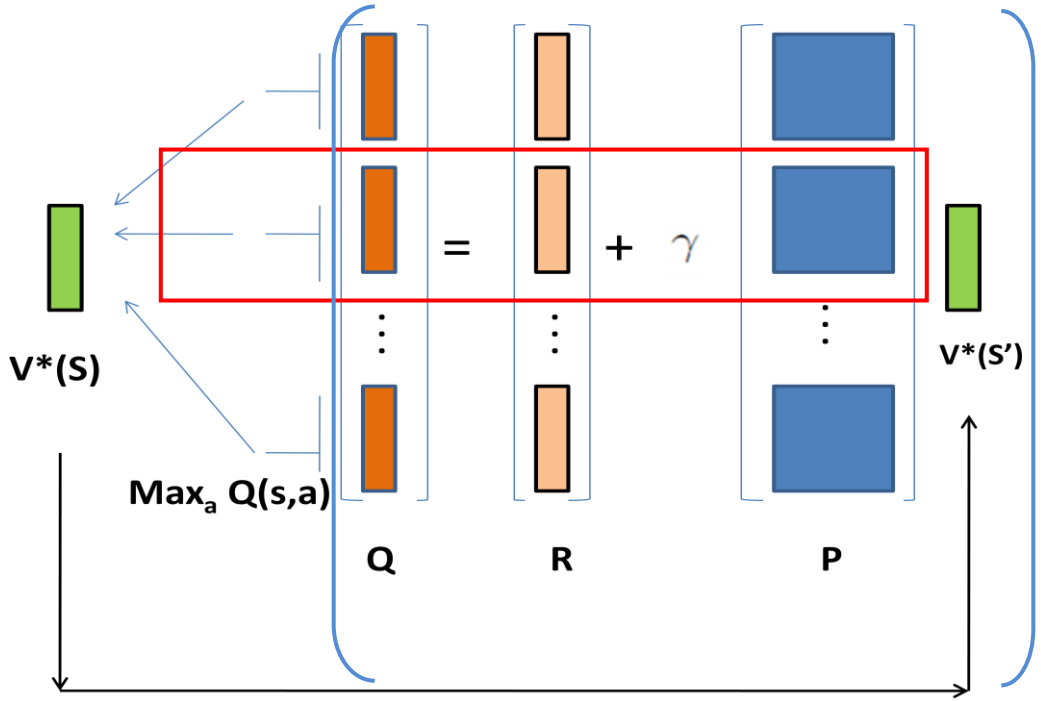
2.3 动态规划：值迭代法

2.4 值迭代方法比较

DP值迭代法

基本思想

$$v(S) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_S^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{S\,s'}^a v(s') \right)$$



对最优策略的优化值进行迭代。

Policy Improvement

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} q_\pi(s, a)$$

DP值迭代法

算法

Three simple ideas for asynchronous dynamic programming:

- *In-place* dynamic programming
- *Prioritised sweeping*
- *Real-time* dynamic programming

算法

In-place dynamic programming

- Synchronous value iteration stores two copies of value function

for all s in $\mathcal{S}$

$$v_{new}(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{old}(s') \right)$$

$$v_{old} \leftarrow v_{new}$$

- In-place value iteration only stores one copy of value function

for all s in $\mathcal{S}$

$$v(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right)$$

算法

Prioritised Sweeping

- Use magnitude of Bellman error to guide state selection, e.g.

$$\left| \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v(s') \right) - v(s) \right|$$

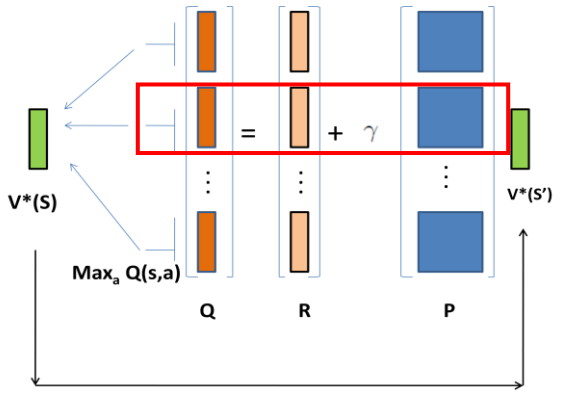
误差导向的更新，例如更新最大误差或满足误差界的 s 对应的 v(s)。

算法

Real-Time Dynamic Programming

- Idea: only states that are relevant to agent
- Use agent's experience to guide the selection of states
- After each time-step S_t, A_t, R_{t+1}
- Backup the state S_t

$$v(S_t) \leftarrow \max_{a \in \mathcal{A}} \left(\mathcal{R}^a_{S_t} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{S_t s'} v(s') \right)$$



思考问题

以上的方法中，DP 环境参数已知，

如果环境参数未知，如何随机逼近等效的 DP 值估计?

第二章 动态规划方法

2.1 Bellman 公式

2.2 动态规划：策略收敛法

2.3 动态规划：值迭代法

2.4 值迭代方法比较

值迭代方法比较

值迭代方法概括

Dynamic Programming (DP)

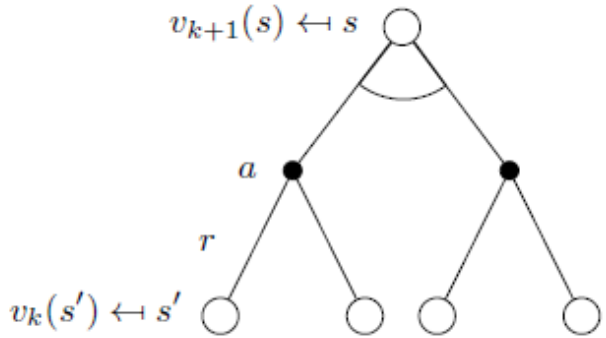
Monte Carlo (MC)

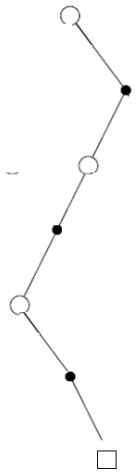
Temporal Difference (TD)

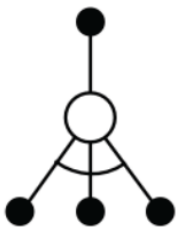
Q-Learning

值迭代方法比较

Backup 比较







D P **MC** **TD** **Q-learning**

本讲参考文献

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. (Second edition, in progress，draft.
2. David Silver，Slides@《Reinforcement Learning:An Introduction》,2016.
3. Simon Haykin，申富饶等译，神经网络与学习机器，第三版。