

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

2018-6-25

第二章 动态规划方法

2.1 Bellman 公式

2.2 Bellman 最优方程

2.3 动态规划法

2.4 值迭代方法

第二章 动态规划方法

2.1 Bellman 公式

2.2 Bellman 最优方程

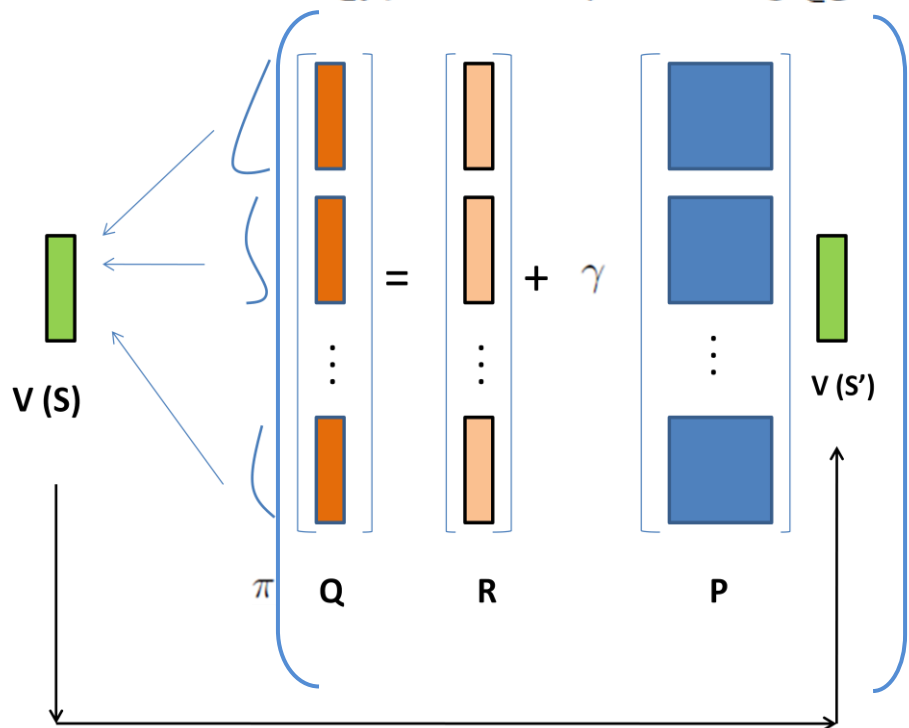
2.3 动态规划法

2.4 值迭代方法

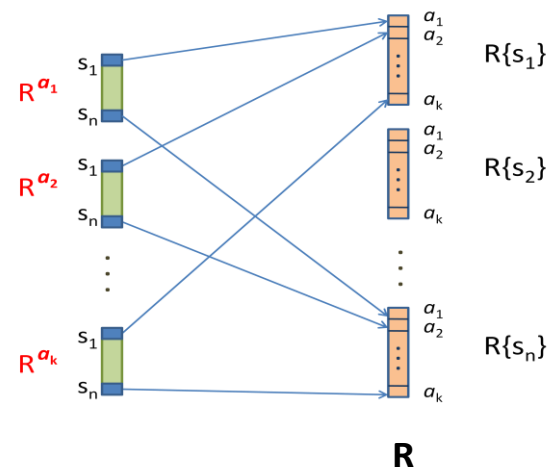
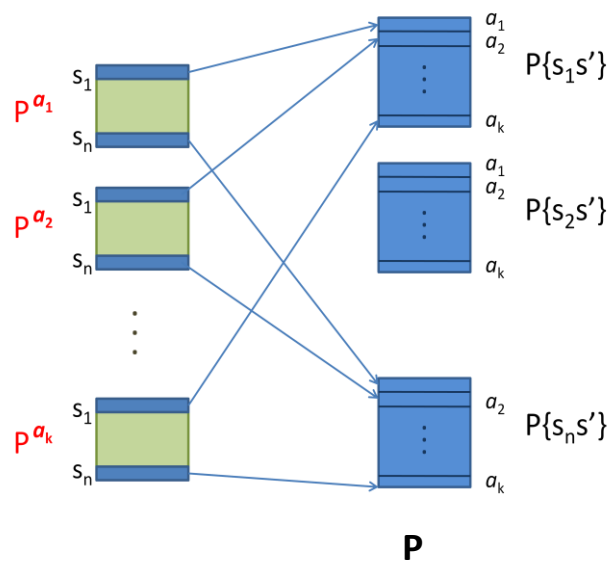
Bellman 公式

Bellman Expectation Equation for V

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



其中,



Bellman 公式

■ Bellman Expectation Equation for V

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

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

The diagram illustrates the Bellman Expectation Equation for V in two forms: a state-action based representation and a state-based representation.

Left Representation (State-Action based):

- A green vertical bar represents the state value $v(s)$.
- A large blue bracket groups the terms π , Q , R , and P .
- π is a vector of orange rectangles representing action probabilities.
- Q is a vector of orange rectangles representing state-action values.
- R is a vector of orange rectangles representing rewards.
- P is a matrix of blue squares representing transition probabilities.
- $v(s')$ is a green vertical bar representing the value of the next state.

Right Representation (State-based):

- A green vertical bar represents the state value $v_{\pi}(s)$.
- A large blue bracket groups the terms π , R , P , and $v_{\pi}(s')$.
- π is a vector of orange rectangles representing action probabilities.
- R is a vector of orange rectangles representing rewards.
- P is a matrix of blue squares representing transition probabilities.
- $v_{\pi}(s')$ is a green vertical bar representing the value of the next state.

The diagram shows the transformation from the state-action based representation to the state-based representation, where the equation is simplified to $v_{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v_{\pi}$.

第二章 动态规划方法

2.1 Bellman 公式

2.2 Bellman 最优方程

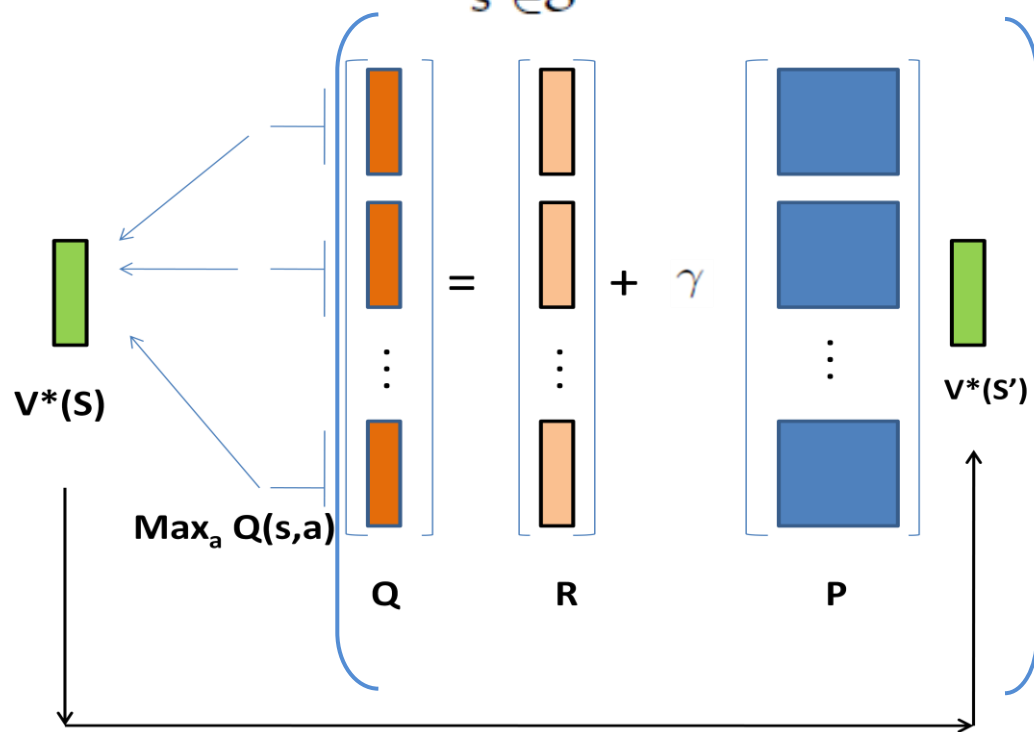
2.3 动态规划法

2.4 值迭代方法

Bellman 最优方程

■ Bellman Optimality Equation for v_*

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



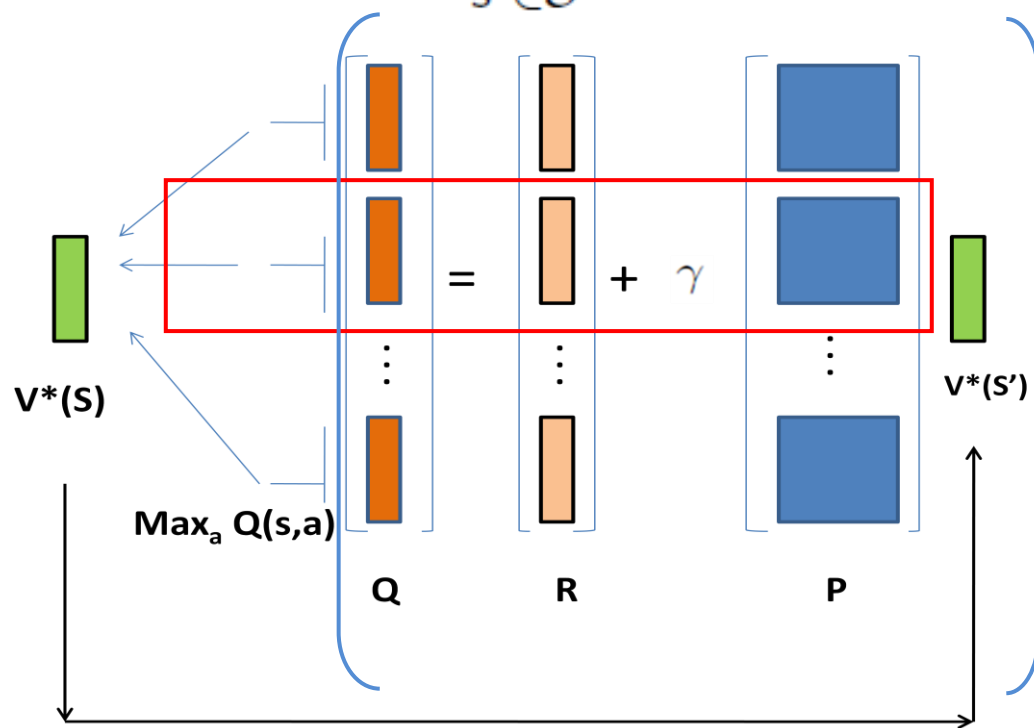
Policy Improvement

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

Bellman 最优方程

■ Bellman Optimality Equation for v_*

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



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

第二章 动态规划方法

2.1 Bellman 公式

2.2 Bellman 最优方程

2.3 动态规划法

2.4 值迭代方法

动态规划法

动态规划

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

- For prediction:

- Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π
- or: MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- Output: value function v_π

Bellman 公式

- Or for control:

- Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- Output: optimal value function v_*
- and: optimal policy π_*

Bellman 最优方程

动态规划法

动态规划

- A method for solving complex problems
- By breaking them down into subproblems
 - Solve the subproblems
 - Combine solutions to subproblems
- For prediction:
 - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and policy π
 - or: MRP $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
 - Output: value function v_π
- Or for control:
 - Input: MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
 - Output: optimal value function v_*
 - and: optimal policy π_*

Bellman 最优方程

动态规划法

DP 策略收敛法

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: $V \rightarrow V_{in}$, 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。

动态规划法

D P 策略收敛法

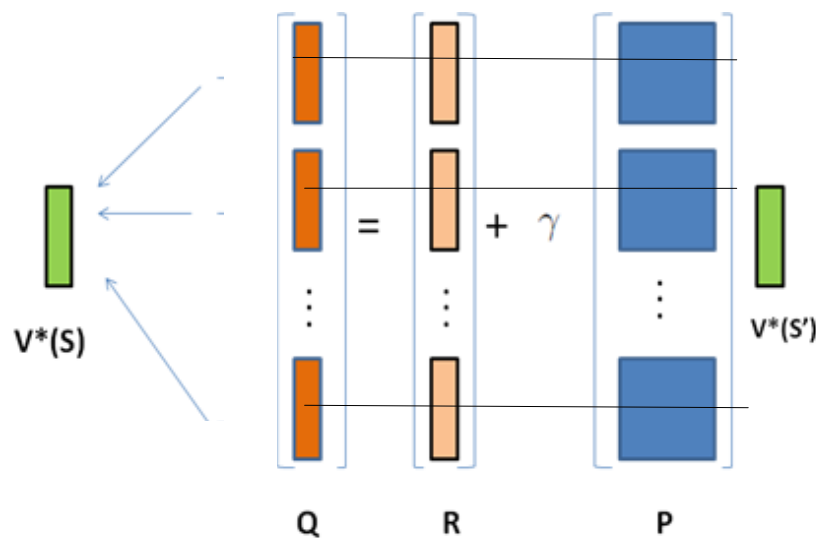
初始化最优策略 u ,

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

Step2: $V \rightarrow V_{in}$, 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。



动态规划法

DP 策略收敛法

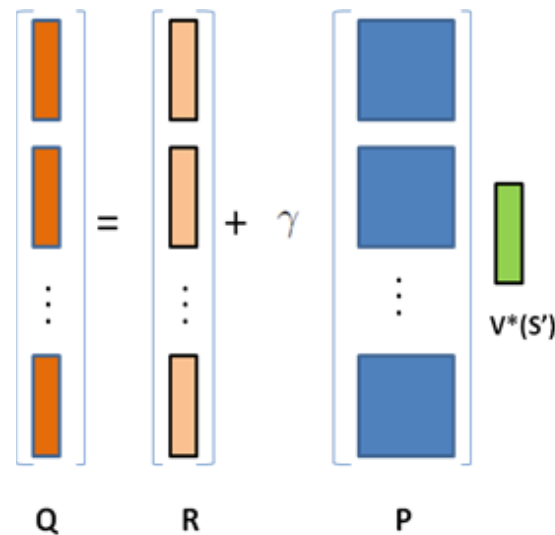
初始化最优策略 u ,

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

Step2: $V \rightarrow V_{in}$, 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。



动态规划法

DP 策略收敛法

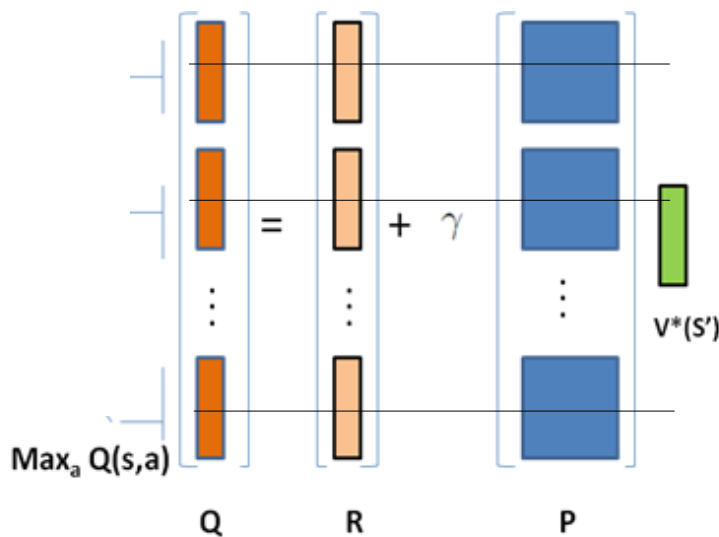
初始化最优策略 u ,

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

Step2: $V \rightarrow V_{in}$, 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。



动态规划法

DP 策略收敛法

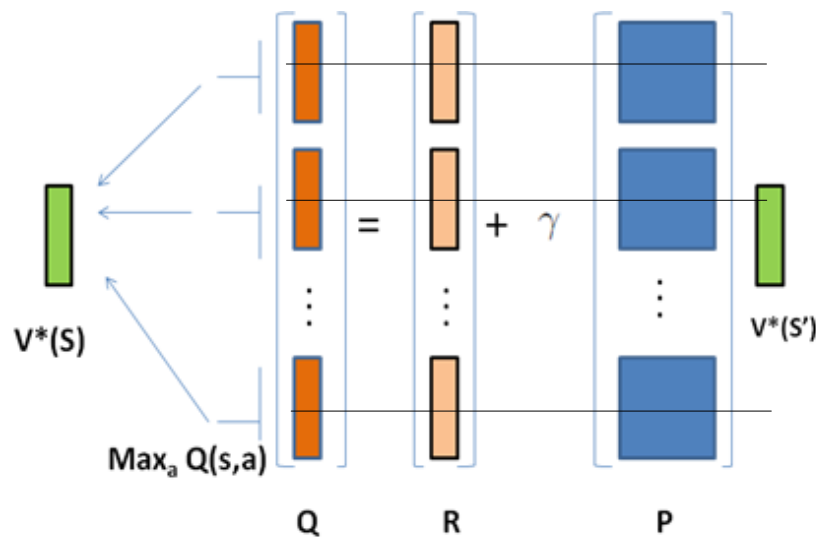
初始化最优策略 u ,

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

Step2: $V \rightarrow V_{in}$, 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。



动态规划法

DP 值迭代法

Three simple ideas for asynchronous dynamic programming:

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

动态规划法

DP 值迭代法

■ 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}

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

Bellman 最优方程

DP 值迭代法

■ 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 $v(s)$ 。

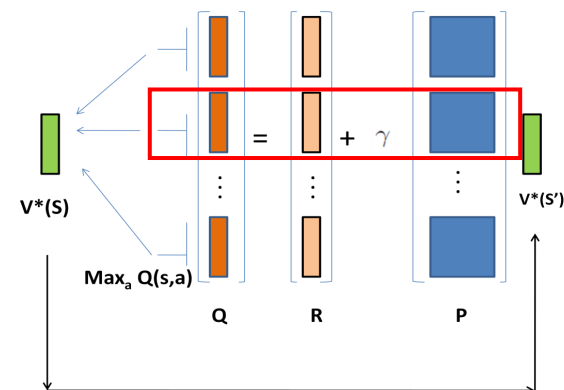
Bellman 最优方程

DP 值迭代法

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

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



第二章 动态规划方法

2.1 Bellman 公式

2.2 Bellman 最优方程

2.3 动态规划法

2.4 值迭代方法

值迭代方法

值迭代方法概括

Dynamic Programming (DP)

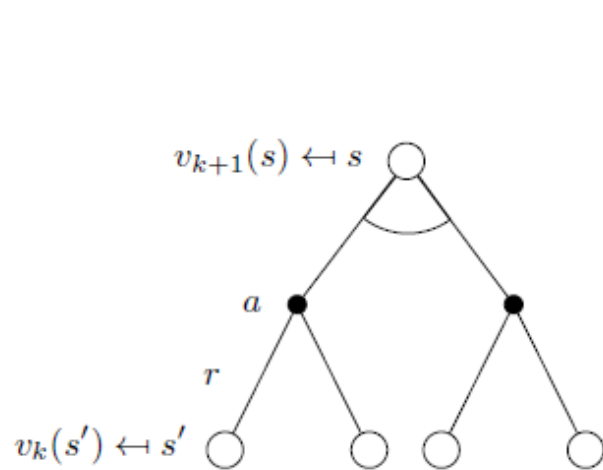
Monte Carlo (MC)

Temporal Difference (TD)

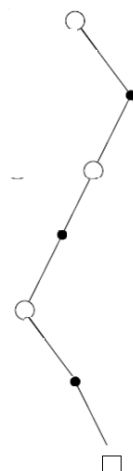
Q-Learning

值迭代方法

Backup 比较



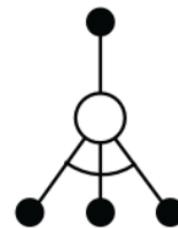
D P



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。