Reinforcement Learning Markov Decision Process Notes

Lu Hong [1]

[1] *Nanjing University of Aeronautics and Astronautics*

September 22, 2019

1 Markov Process

"The future is independent of the past given the present". A state S_t is *Markov* iif.

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, ..., S_t]$$

This requires environment to be fully observed. Markov Process (or Markov Chain) is a *memoryless random process*, represented by a tuple $< S, P >$

1.1 Markov Chain

suppliment from WangCai's Note

Let $s_t \in S$, and if S is countable, we call the Markov Process with this countable state space Markov Chain.

2 Markov Reward Process

Markov Reward Process is Markov Process with values, represented by a tuple $< S, P, R, \gamma >$, where $R = \mathbb{E}[R_{t+1}|S_t = s]$ and γ is a discount factor.

Return G_t is the *total discounted reward* at time-step t presented by

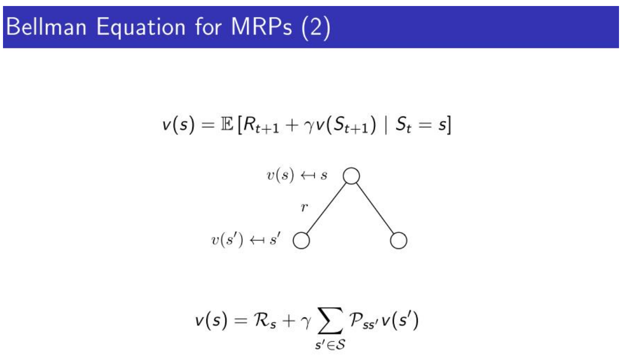
$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

Also, we define Value Function to indicate the long-term value of the state s.

$$V(s) = \mathbb{E}[G_t|S = s]$$

2.1 Bellman Equation

* Bellman Equation for MRPs, it demonstrate that MRPs can be presented in recursive format.



$$\begin{aligned} v(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ...)|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s] \end{aligned}$$

To express Bellman Equation in matrix form as

$$v = R + \gamma P v,$$

we can solve Bellman Equation easily as linear equation.

$$\begin{aligned} v &= R + \gamma P v \\ (I - \gamma P)v &= R \\ v &= (I - \gamma P)^{-1} R \end{aligned}$$

•

3 Markov Decision Process

A MDP is a MRP with decisions, represented by a tuple $< S, A, P, R, \gamma >$, to be specific, some params have been changed after action is introduced.

$$P_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$
$$R_{s}^{a} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

3.1 Policies

Policies: A policy π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

- A policy fully characterizes the agent's behaviour
- Policy is what we want in RL problem
- Policy is only asscoiated with current state
- Policy is static if it's certain
- Agent can update policy during the time
- If π is one-hot, then policy is certain

Policies (2)

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, ...$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^{\pi} \rangle$
- The state and reward sequence $S_1, R_2, S_2, ...$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$
- where

$$\mathcal{P}_{s,s'}^{\pi} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^{a}$$
$$\mathcal{R}_{s}^{\pi} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_{s}^{a}$$

After we introduce Policy, we should reinvent the value function. the state-value funcion is:

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$$

the action-value function is:

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$$

3.2 Bellman Expectation Equation

Bellman Expectation Equation

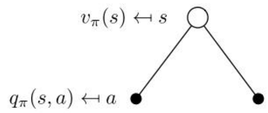
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_{\pi}(s) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s\right]$$

The action-value function can similarly be decomposed,

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

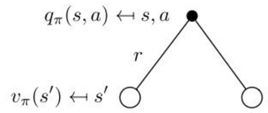
Bellman Expectation Equation for V^{π}



$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_{\pi}(s, a)$$

For each action nod, q-value is generated the same way.

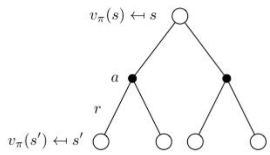
Bellman Expectation Equation for Q^{π}



$$q_{\pi}(s, a) = \mathcal{R}_{s}^{a} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^{a} v_{\pi}(s')$$

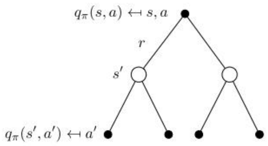
Two process combined in various order, we can get Bellman Expectation Equation for v_{π} and q_{π}

Bellman Expectation Equation for v_{π} (2)



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

Bellman Expectation Equation for q_π (2)



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

3.3 Optimal Value Function

3.3.1 Definition

The optimal state-value function $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

3.3.2 Optimal Policy

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$*
- *All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$*
- *All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) = q_*(s,a)$*

3.3.3 Find Optimal Policy

└ Optimal Value Functions

Finding an Optimal Policy

An optimal policy can be found by maximising over $q_*(s,a)$,

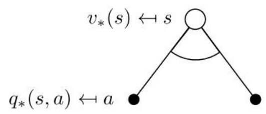
$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s,a)$, we immediately have the optimal policy

3.4 Bellman Optimality Equation

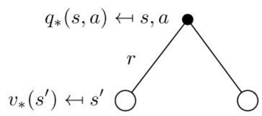
Bellman Optimality Equation for v_*

The optimal value functions are recursively related by the Bellman optimality equations:

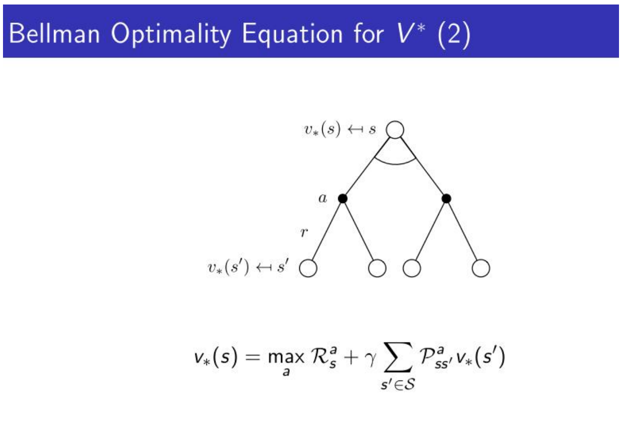


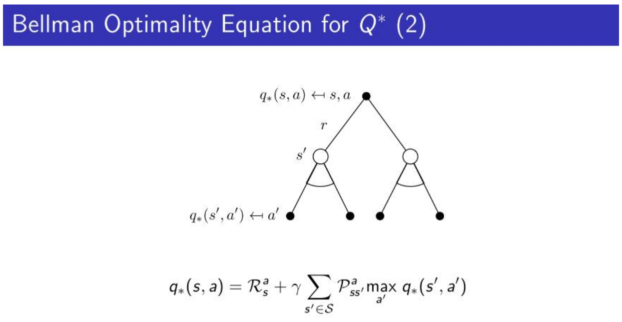
$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equation for Q^*



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



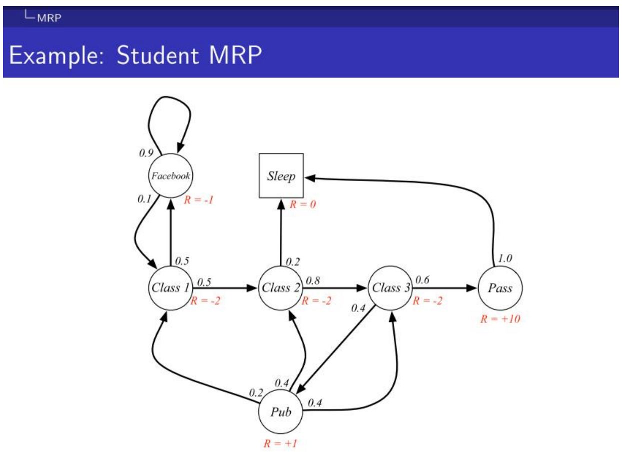


4 Example Demonstration

I import example from (Silver, 2018) and explanation from (WangCai, 2019), accompanied with my personal perspective to finish this part.

4.1 MRP Example

Given the following Markov Reward Process



, We can extract this process into a matrix:

$$\begin{bmatrix} \textbf{Reward} & -1 & -2 & -2 & -2 & 10 & 1 & 0 \\ \textbf{State} & FaceBook & Class1 & Class2 & Class3 & Pass & Pub & Sleep \\ FaceBook & 0.9 & 0.1 & & & & & \\ Class1 & 0.5 & & 0.5 & & & & \\ Class2 & & & & 0.8 & & & 0.2 \\ Class3 & & & & & 0.6 & 0.4 & \\ Pass & & & & & & & 1 \\ Pub & & 0.2 & 0.4 & 0.4 & & & \\ Sleep & & & & & & & 1 \end{bmatrix}$$

where

$$\mathcal{S} = \begin{bmatrix} FaceBook \\ Class1 \\ Class2 \\ Class3 \\ Pass \\ Pub \\ Sleep \end{bmatrix},$$

$$\mathcal{P} = \begin{bmatrix} 0.9 & 0.1 & & & & & \\ 0.5 & & 0.5 & & & & \\ & & & 0.8 & & & 0.2 \\ & & & & 0.6 & 0.4 & \\ & & & & & & 1 \\ & 0.2 & 0.4 & 0.4 & & & \\ & & & & & & 1 \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} -1 \\ -2 \\ -2 \\ -2 \\ 10 \\ 1 \\ 0 \end{bmatrix},$$

$$\gamma \in [0, 1]$$

where we can use these params to calculate state value $v(s)$ using bellman equation

$$v = (I - \gamma P)^{-1} R$$

$$= \left(\begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} - \gamma \begin{bmatrix} 0.9 & 0.1 & & & & & \\ 0.5 & & 0.5 & & & & \\ & & & 0.8 & & & 0.2 \\ & & & & 0.6 & 0.4 & \\ & & & & & & 1 \\ & 0.2 & 0.4 & 0.4 & & & \\ & & & & & & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} -1 \\ -2 \\ -2 \\ -2 \\ 10 \\ 1 \\ 0 \end{bmatrix}$$

The result state-value vector is stored in s, shown in 1

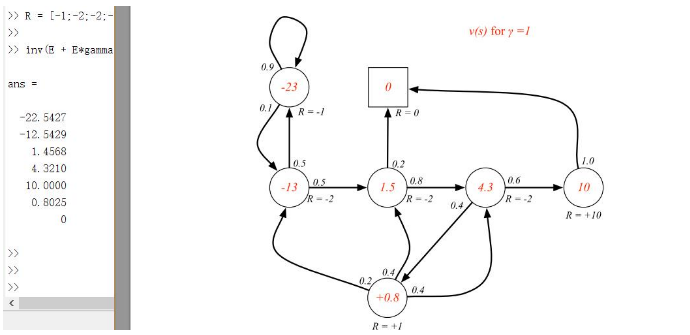
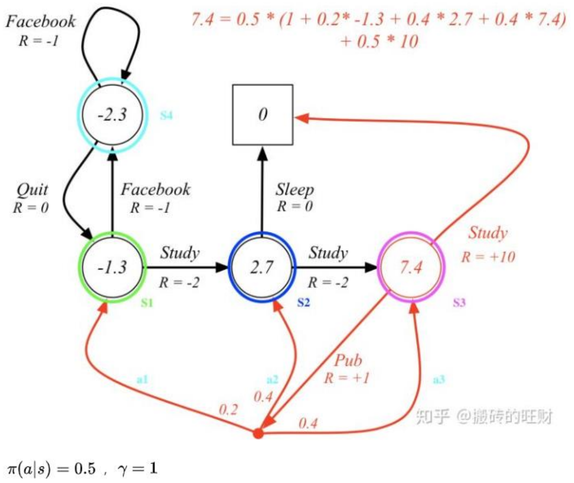


Figure 1: *MRP results*

4.2 MDP Example

Introduce MDP example from the lecture.



First, we can use state-value function to calculate the value of each state. Using

$$v_\pi(s) = \sum_{a\in A} \pi(a|s)(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a v_\pi(s^{'}))$$

since $\pi(a|s) = 0.5$, it means the decision is uniformly random. Also, the state value is initialized as zero-vector actually. for $v(1)$,

$$v(1) = 0.5 * (-1 + v(4)) + 0.5 * (-2 + v(2))$$

for $v(2)$,

$$v(2) = 0.5 * (-2 + v(3))$$

for $v(3)$,

$$v(3) = 0.5*(10+0)+0.5*(1+(0.2*v(1)+0.4*v(2)+0.4*v(3)))$$

for $v(4)$,

$$v(4) = 0.5 * (-1 + v(4)) + 0.5 * (0 + v(1))$$

From these four equations, we can calculate four state values.

$$\begin{bmatrix} v1 \\ v2 \\ v3 \\ v4 \end{bmatrix} = \begin{bmatrix} -1.3 \\ 2.7 \\ 7.4 \\ -2.3 \end{bmatrix}$$

After we have got state value, we can use the same method to obtain the value of action.

Reference

Silver, D. (2018). “Reinforcement Learning”. In: *Teaching*. URL: `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`.

WangCai (2019). In: URL: `https://zhuanlan.zhihu.com/p/50685812`.