

Notes for RL

Jinze LI

June 13, 2024

Abstract

Notes

1 Outline

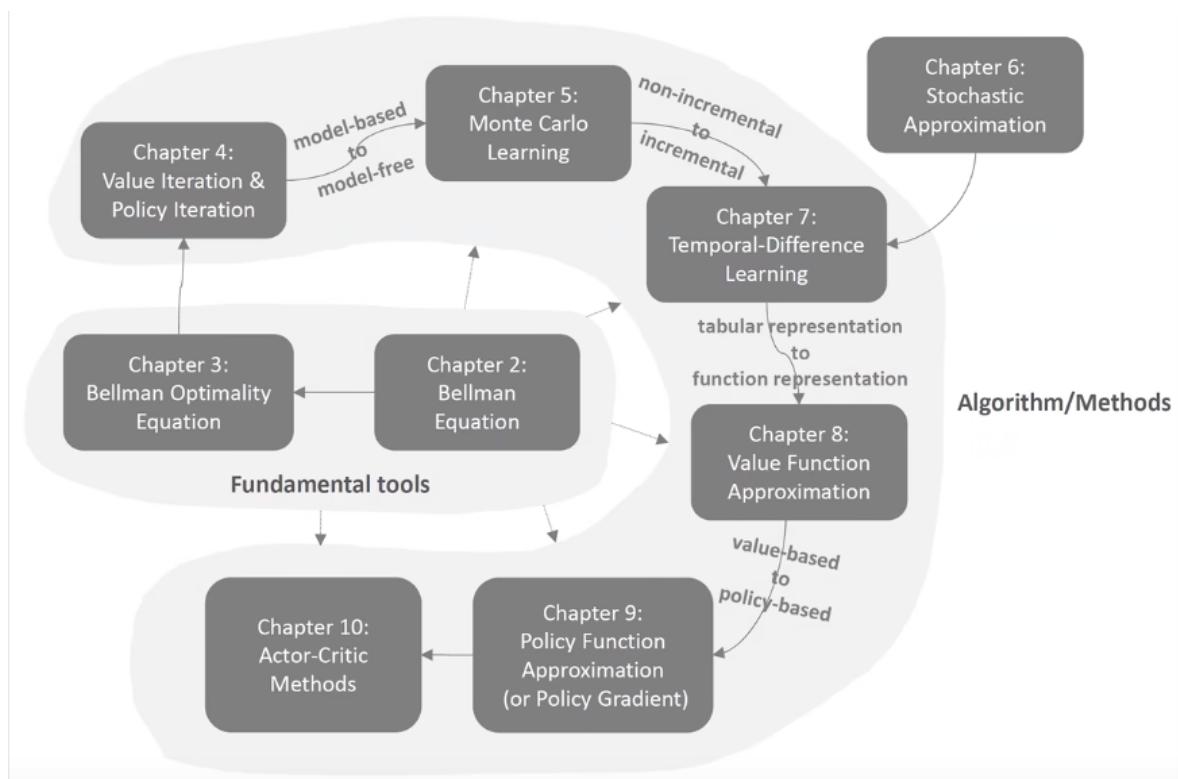


Figure 1: Enter Caption

2 Value Function Iteration

2.1 MP (Markov Process)

$$V(s) = E[G_t | s_t = s] = E[R_{t+1} + \gamma R_{t+2} + \dots | s_t = s] \quad (1)$$

$$= E[R_{t+1} + \gamma V(s') | s_t = s] \quad (2)$$

$$= R(s) + \gamma \sum_{s' \in S} P(s'|s)V(s') \quad (3)$$

(4)

2.2 MRP (Markov Reward Process)

2.3 MDP (Markov Decision Process)

MDP = MRP + Action

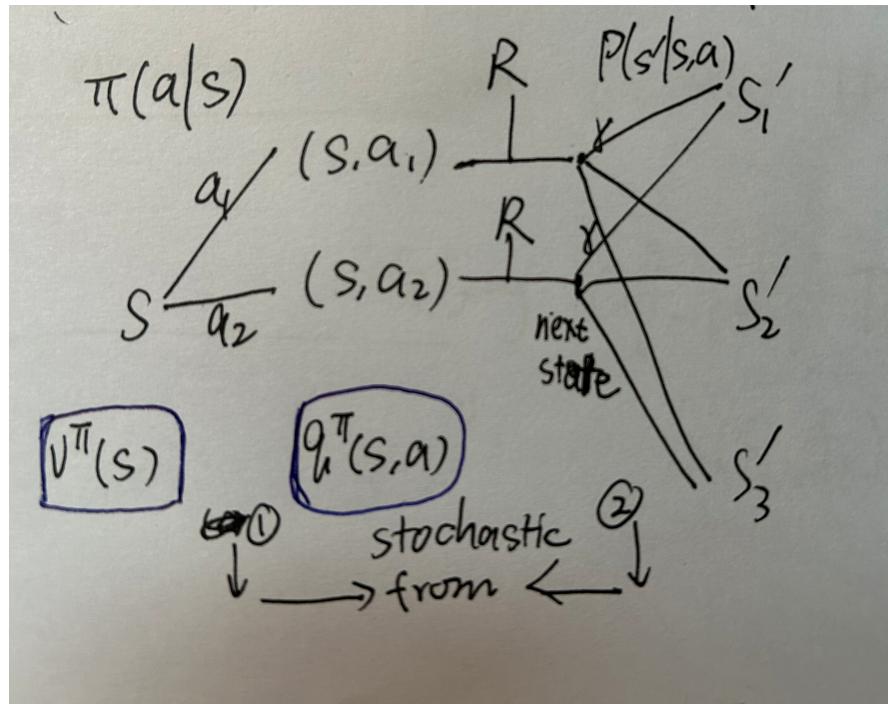


Figure 2: Enter Caption

$$v^\pi(s) = E_\pi[G_t | s_t = s] \quad (5)$$

$$q^\pi(s, a) = E_\pi[G_t | s_t = s, A_t = a] \quad (6)$$

(7)

Relation

$$v^\pi(s) = \sum_{a \in A} \pi(a|s) q^\pi(s, a) \quad (8)$$

$$q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \quad (9)$$

$$v^\pi(s) = E_\pi[R_{t+1} + \gamma v^\pi(s_{t+1}) | s_t = s] \quad (10)$$

$$q^\pi(s, a) = E_\pi[R_{t+1} + \gamma q^\pi(s_{t+1}, a_{t+1}) | s_t = s, A_t = a] \quad (11)$$

$$v^\pi(s) = \sum_{a \in A} \pi(a|s) (R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v^\pi(s')) \quad (12)$$

$$q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \sum_{a' \in A} \pi(a'|s') q^\pi(s', a') \quad (13)$$

3 Policy Iteration

4 Mean Estimation

5 on/off- line learning

6 MC

first visit first, every visit

All the algorithms we introduced in this lecture can be expressed in a unified expression:

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)[q_t(s_t, a_t) - \bar{q}_t],$$

where \bar{q}_t is the *TD target*.

Different TD algorithms have different \bar{q}_t .

Algorithm	Expression of \bar{q}_t
Sarsa	$\bar{q}_t = r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$
n -step Sarsa	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^n q_t(s_{t+n}, a_{t+n})$
Expected Sarsa	$\bar{q}_t = r_{t+1} + \gamma \sum_a \pi_t(a s_{t+1}) q_t(s_{t+1}, a)$
Q-learning	$\bar{q}_t = r_{t+1} + \gamma \max_a q_t(s_{t+1}, a)$
Monte Carlo	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \dots$

The MC method can also be expressed in this unified expression by setting $\alpha_t(s_t, a_t) = 1$ and hence $q_{t+1}(s_t, a_t) = \bar{q}_t$.

Figure 3: Enter Caption

All the algorithms can be viewed as stochastic approximation algorithms solving the Bellman equation or Bellman optimality equation:

Algorithm	Equation aimed to solve
Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) S_t = s, A_t = a]$
n -step Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n q_\pi(s_{t+n}, a_{t+n}) S_t = s, A_t = a]$
Expected Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{A_{t+1}}[q_\pi(S_{t+1}, A_{t+1})] S_t = s, A_t = a]$
Q-learning	BOE: $q(s, a) = \mathbb{E}[R_{t+1} + \max_a q(S_{t+1}, a) S_t = s, A_t = a]$
Monte Carlo	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots S_t = s, A_t = a]$

Figure 4: Enter Caption

7 TD

7.1 on/off-policy

The Behavior Policy is used to generate experience samples.

The Target Policy is toward our optimal policy.

On-policy learning: the above two are the same.

Off-policy learning: the above two are NOT the same. (Learn Experience from Others.)

eg. Set behavior policy to be exploration policy, and we use the experience to do the target policy.

7.2 examples

Sarsa is on-policy. MC is on-policy

Q-learning is on/off-policy.

Sarsa is on-policy.

- First, Sarsa aims to solve the Bellman equation of a given policy π :

$$q_\pi(s, a) = \mathbb{E} [R + \gamma q_\pi(S', A')|s, a], \quad \forall s, a.$$

where $R \sim p(R|s, a)$, $S' \sim p(S'|s, a)$, $A' \sim \pi(A'|S')$.

- Second, the algorithm is

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t) \left[q_t(s_t, a_t) - [r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})] \right],$$

which requires $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$:

- If (s_t, a_t) is given, then r_{t+1} and s_{t+1} do not depend on any policy!
- a_{t+1} is generated following $\pi_t(s_{t+1})$!
- π_t is both the target and behavior policy.

$\pi_t \rightarrow \text{exp} \rightarrow q_{\pi_t}$

Figure 5: Enter Caption

Q-learning tabular version

Q-Learning (tabular version)

- Observe a transition (s_t, a_t, r_t, s_{t+1}) .
- TD target: $y_t = r_t + \gamma \max_a Q^*(s_{t+1}, a)$.

	Action a_1	Action a_2	Action a_3	Action a_4	...
State s_1					
State s_2					
State s_3					
:					

Figure 6: Enter Caption

functional approximation: Linear

Deep learning (DQN)

Q-Learning (DQN Version)

- Observe a transition (s_t, a_t, r_t, s_{t+1}) .
- TD target: $y_t = r_t + \gamma \cdot \max_a Q(s_{t+1}, a; \mathbf{w})$.
- TD error: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$.
- Update: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial Q(s_t, a_t; \mathbf{w})}{\partial \mathbf{w}}$.

Figure 7: Enter Caption

8 Value Function Approximation

9 Policy Function Approximation (Policy Gradient)

10 ref

ZHAO shiyu course

le yuan blog

Liang tengyuan, course

SDSC8006, Clint Chin Pang Ho

References