Intro to RL

Xudong Wang

[1]Department of Industrial & Systems Engineering, The University of Tennessee, Knoxville.

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Definition

- The State Value Function, denoted as $V(s)$, is a fundamental concept in reinforcement learning. It represents the expected cumulative reward an agent can obtain from a given state s while following a specific policy.
- Mathematical Representation:

$$V^{\pi}(s) = E_{\pi}[\mathrm{G_t} | \mathrm{S_t} = \mathrm{s}] = \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \forall s \in \mathcal{S}$$

Understanding State Value Function

- $V(s)$ quantifies the goodness of a state s under a given policy π.
- It indicates how desirable it is to be in state s when following policy π.

State Value Function is used for:

- Evaluating the quality of a state under a policy.
- Solving reinforcement learning problems using methods like Value Iteration and Policy Iteration.

Properties of State Value Function

- **Markov Property:** $V(s)$ depends only on the current state s and the policy π.
- **Recursive Nature:** $V(s)$ is defined in terms of itself, involving future rewards.
- **Optimal State Values:** There exists an optimal policy π^* such that $V^*(s) = V^{\pi^*}(s)$, where $V^*(s)$ is the state value function for the optimal policy.

Computing State Value Function

- Value Iteration: An iterative method for finding the optimal $V^*(s)$.
- Policy Evaluation: Part of the Policy Iteration process.

Example: Grid World

- Consider a 5*5 grid world with states.
- An agent can move up, down, left, or right.
- Every move will have a -1 reward.
- When entering location A, it will have a +10 reward (no move cost).
- When entering location B, it will have a -2 reward (no move cost). A is the terminal point.
- The goal is to reach the terminal state with maximum cumulative reward.

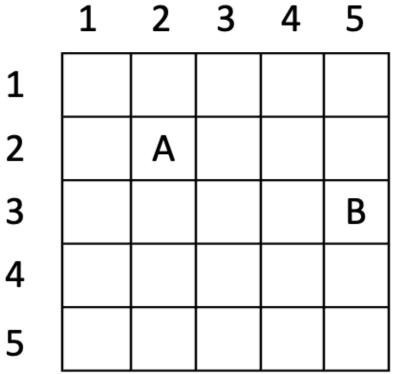

Figure: Grid World Example

COMPUTING STATE VALUE FUNCTION

	1	2	3	4	5
1	0	0	0	0	0
2	0	T	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

$\longrightarrow$

	1	2	3	4	5
1	-1	10	-1	-1	-1
2	10	T	10	-1	-1
3	-1	10	-1	-1	-1
4	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1

π: *greedy*
γ: 0.9

$$\begin{aligned} V(1,2) &= \max\big(E(left), E(right), E(down)\big) \\ &= \max(-1+\gamma V(1,1), -1+\gamma V(3,1), 10) \\ &= 10 \end{aligned}$$

$$\begin{aligned} V(3,4) &= \max\big(E(left), E(right), E(down), E(up)\big) \\ &= \max\big(-1+\gamma V(3,3), -2+\gamma V(3,5), -1+\gamma V(4,4), -1+\gamma V(2,4)\big) \\ &= -1 \end{aligned}$$

FIGURE: State Value Function for Grid World

Computing State Value Function

	1	2	3	4	5
1	0	0	0	0	0
2	0	T	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

$\longrightarrow$

	1	2	3	4	5
1	-1	2.67	-1	-1	-1
2	2.67	T	2.33	-1	-1.34
3	-1	2.33	-1	-1.25	-1
4	-1	-1	-1	-1	-1.34
5	-1	-1	-1	-1	-1

$\pi: random$

$\gamma: 0.9$

$$\begin{aligned} V(1,2) &= \frac{1}{3} * (E(left) + E(right) + E(down)) \\ &= \frac{1}{3} * (-1 + \gamma V(1,1) - 1 + \gamma V(3,1) + 10) \\ &= \frac{8}{3} \end{aligned}$$

$$\begin{aligned} V(3,4) &= \frac{1}{4} * (E(left) + E(right) + E(down) + E(up)) \\ &= \frac{1}{4} * (-1 + \gamma V(3,3) - 2 + \gamma V(3,5) - 1 + \gamma V(4,4) - 1 + \gamma V(2,4)) \\ &= -1.25 \end{aligned}$$

Action Value Function (Q-Function)

- The Action Value Function, denoted as $Q(s, a)$, represents the expected cumulative reward an agent can obtain from taking action a in state s and following a specific policy.
- Mathematical Representation:

$$Q^{\pi}(s,a) = E_{\pi}[\mathrm{G_t}|\mathrm{S_t} = \mathrm{s}, A_t = a] = \mathrm{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right], \forall\, s \in \mathcal{S}$$

Understanding Action Value Function

- $Q(s, a)$ quantifies the goodness of taking a specific action a in a given state s under a particular policy π.
- It indicates the expected return if the agent selects action a while in state s and continues to follow policy π.

Action Value Function is used for:

- Evaluating the quality of actions in specific states.
- Selecting the best action to maximize the expected return.

Relationship Between State and Action Value Functions

Connection:
The State Value Function $V(s)$ based on greedy method and Action Value Function $Q(s,a)$ are related as follows:

$$V(s) = \max_{a} Q(s,a)$$

Optimal Policy: For the optimal policy π^*:

$$\pi^*(s) = \arg\max_{a} Q^*(s,a)$$

where $Q^*(s,a)$ is the action value function for the optimal policy.

Similar to State Value Function, Action Value Function can be computed using methods like Value Iteration and Policy Iteration.

Computing Action Value Function

Based on a random policy:

$$Q^{\pi}((0,0), right) = \sum_{s'} p(s', r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s', a')]$$
$$= \sum_{s'} p(s', r|s,a)[r + \gamma V^{\pi}(s')]$$

$s' = (0,0),\ r = -1,\ \gamma = 0.9$

Computing Action Value Function

Consider another example: The agent is in state 1. When it takes action a, it will get a reward α and reach state 2 or 3 with an equal probability. When it takes action b, it will have a 0.3 probability of getting a reward β and reaching state 3; and a 0.7 probability of getting a reward ν and reaching state 4.

$$Q^{\pi}(1,a) = \sum_{s'} p(s',r|1,a)[r+\gamma V^{\pi}(s')]$$
$$= 0.5 * [\alpha+\gamma V^{\pi}(2)] + 0.5 * [\alpha+\gamma V^{\pi}(3)]$$

$$Q^{\pi}(1,b) = \sum_{s'} p(s',r|1,a)[r+\gamma V^{\pi}(s')]$$
$$= 0.3 * [\beta+\gamma V^{\pi}(3)] + 0.5 * [\nu+\gamma V^{\pi}(4)]$$

Dynamic programming

Dynamic programming is the collection of algorithms that can be used to compute optimal policies given perfect model of the environment.

- DP constitutes a theoretically optimal methodology
- In reality it is often limited since DP is computationally expensive.
- Key Idea (In general): Use value function to derive optimal policy
- Need to find a way to calculate optimal state value function

Value Iteration

Definition:
Value Iteration is an iterative algorithm used to find the optimal state value function $V^*(s)$ and, consequently, the optimal policy π^* in reinforcement learning.

- It starts with an initial value function and repeatedly updates it until it converges to the optimal value function.
- The optimal policy can be derived directly from the optimal value function.

The Value Iteration algorithm works as follows:

- Initialize $V(s)$ arbitrarily for all states.
- Iterate until convergence:
  - Update each state's value using the Bellman optimality equation:

$$V(s) \leftarrow \max_a \sum_{s',r} P(s', r|s, a)[r + \gamma V(s')]$$

- The resulting $V^*(s)$ represents the optimal state values.

Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a)\left[r + \gamma V(s')\right]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r | s, a)\left[r + \gamma V(s')\right]$

Policy Iteration

Definition Policy Iteration is an algorithm used to find the optimal policy in reinforcement learning. It alternates between two steps: policy evaluation and policy improvement.

- In the policy evaluation step, it evaluates the current policy to get its value function $V^\pi(s)$.
- In the policy improvement step, it improves the policy based on the value function.
- It repeats these steps until the policy converges to the optimal policy.

The Policy Iteration algorithm works as follows:

- Initialize a policy π arbitrarily.
- Iterate until convergence:
  - Policy Evaluation: Compute the value function $V^\pi(s)$ for the current policy.
  - Policy Improvement: Update the policy based on the current value function:

$$\pi(s) \leftarrow \arg\max_a \sum_{s',r} P(s',r|s,a)[r + \gamma V^\pi(s')]$$

- The resulting policy π^* is the optimal policy.

Policy Iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\left[r + \gamma V(s')\right]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Value Iteration vs. Policy Iteration

Comparison

- **Value Iteration:**
  - More straightforward and computationally efficient.
  - Directly computes the optimal state values and policy.
  - May require more iterations to converge.
- **Policy Iteration:**
  - Faster convergence in practice.
  - Alternates between policy evaluation and policy improvement.
  - Requires solving a linear system of equations in the policy evaluation step.

Choice of Method The choice between Value Iteration and Policy Iteration often depends on the specific problem, computational resources, and the need for a fast or efficient solution.

Comparison between DP and MC

DP

- theoretically optimal result
- expensive computation
- model-based

MC

- can be "just as well" as DP
- less computation and memory
- model free

Monte Carlo Policy Evaluation

- We still need to estimate V(s). Estimate from experience
  - Average all returns observed after visiting a given state.
  - Each occurrence of state s in an episode is called a visit to s
- Generate an episode following π:
  $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T$
- First visit MC: Method averages just the returns following first visits to s
- Every Visit MC: Method averages the returns following all the visits to s

Monte Carlo Policy Evaluation

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Example of Value Iteration

Converge after 6 iterations:

π: *greedy*
γ: 0.9

	1	2	3	4	5
1	0	0	0	0	0
2	0	T	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

	1	2	3	4	5
1	-1	10	-1	-1	-1
2	10	T	10	-1	-1
3	-1	10	-1	-1	-1
4	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1

	1	2	3	4	5
1	8	10	8	-1.9	-1.9
2	10	T	10	8	-1.9
3	8	10	8	-1.9	-1.9
4	-1.9	8	-1.9	-1.9	-1.9
5	-1.9	-1.9	-1.9	-1.9	-1.9

...

	1	2	3	4	5
1	8	10	8	6.2	4.58
2	10	T	10	8	6.2
3	8	10	8	6.2	4.58
4	6.2	8	6.2	4.58	3.12
5	4.58	6.2	4.58	3.12	1.81

Example of Policy Iteration

Converge after 4 iterations:

$\gamma: 0.9$

π_0

	1	2	3	4	5
1	←	←	↓	→	↓
2	↑	T	↑	→	↓
3	↑	→	←	↓	←
4	→	↓	↓	→	←
5	→	→	←	←	←

V_0

	1	2	3	4	5
1	0	-1	0	-1	0
2	-1.9	T	-10	-1.9	-1
3	-1	-10	-10	-2.71	-3.44
4	0	-10	-10	-10	-10
5	-1	-10	-10	-10	-10

π_1

	1	2	3	4	5
1	→	↓	←	←	←
2	→	T	←	→	↑
3	↓	↑	→	↑	↑
4	↑	←	←	↑	↑
5	↑	←	←	←	←

V_1

	1	2	3	4	5
1	8	10	8	6.2	4.58
2	10	T	10	1.81	3.12
3	-10	10	-0.43	0.63	1.81
4	-10	-10	-1.39	-0.43	-0.37
5	-10	-10	-10	-1.39	-1.33

...

π_*

	1	2	3	4	5
1	→	↓	←	←	←
2	→	T	←	←	←
3	→	↑	←	←	←
4	→	↑	←	←	←
5	→	↑	←	←	←

V_*

	1	2	3	4	5
1	8	10	8	6.2	4.58
2	10	T	10	8	6.2
3	8	10	8	6.2	4.58
4	6.2	8	6.2	4.58	3.12
5	4.58	6.2	4.58	3.12	1.81