Deep Reinforcement Learning Spring 2020 Assignment 1

Minghao Zhang

1 Optimal Policy for Simple MDP

(a)

According to the definition of the value iteration, we have

$$\begin{aligned} V^0(G) &= 1 \\ V^1(G) &= 1 + \gamma \\ &... \\ V(G) &= \lim_{n \to +\infty} V^n(G) = 1 + \gamma + \gamma^2 + ... = \frac{1}{1-\gamma} \end{aligned}$$

$$\begin{aligned} V(s_{n-1}) &= 0 + \gamma V(G) = \frac{\gamma}{1-\gamma} \\ V(s_{n-2}) &= 0 + \gamma V(s_{n-1}) = \frac{\gamma^2}{1-\gamma} \\ &... \\ V(s_1) &= 0 + \gamma V(s_2) = \frac{\gamma^{n-1}}{1-\gamma} \end{aligned}$$

(b)

We assume that $\gamma \geq 0$ according to the definition of discount factor. When $\gamma \geq 0$, we have

$$\frac{\gamma^t}{1-\gamma} \geq \frac{\gamma^{t+1}}{1-\gamma},\ \forall t \geq 0$$

so the following inequality holds for all cases

$$V(s_{t+1}) \geq V(s_t),\ \forall t \geq 0$$

which means optimal policy will always be choosing a_0 no matter which state we are in. Note that equality holds if and only if $\gamma = 0$, so when it becomes 0, optimal solution won't be unique.

(c)

Intuitively, we can regard the $r_1, ..., r_n$ as a group of basis and all value of value functions are in the generated linear space. So translation on all elements in basis doesn't change the relative size relation between the values, since all values are just added by a same constant.

100th Conference on Neural Information Processing Systems (NeurIPS 2086), Beijing, China.

Specifically, let the new value function for state s be $V'(s)$. Since

$$\begin{aligned} V'(s) &= \sum_{i=1}^{+\infty} \gamma^i (c + r_i) \\ &= V(s) + \frac{c}{1-\gamma} \end{aligned}$$

(d)

Similarly, we have

$$\begin{aligned} V'(s) &= \sum_{i=1}^{+\infty} \gamma^i a(c + r_i) \\ &= aV(s) + \frac{ac}{1-\gamma} \end{aligned}$$

When $a \geq 0$, the optimal policy won't change and $a = 0$ will cause the non-uniqueness. If $a < 0$, the optimal policy will be that agent never achieves the goal, inversely.

2 Running Time of Value Iteration

(a)

Since after reaching s_1, we can only spin in the same place and get 1 i each time step, so we have

$$V = 0 + \sum_{i=1}^{+\infty} r^i = \frac{\gamma}{1-\gamma}$$

(b)

Same as (a), we have

$$V = \frac{\gamma^2}{1-\gamma} + 0 + ... + 0 + ... = \frac{\gamma^2}{1-\gamma}$$

So a_1 is optimal.

(c)

We have

$$V_{n+1}(s_1) = 1 + \gamma V_n(s_1),\ Q_{n+1}(s_0, a_1) = \gamma V_n(s_1)$$

then

$$Q_{n+1} = \gamma(1 + \gamma + ... + \gamma^{n-1} + \gamma^n \times 0) = \gamma \frac{1-\gamma^n}{1-\gamma}$$

And notice that $Q_i(s_0, a_2) = 0,\ \forall i \geq 0$. While $Q(s_0, a_2) > Q(s_0, a_1)$, we will choose the sub-optimal action, which means

$$\gamma \frac{1-\gamma^n}{1-\gamma} > \frac{\gamma^2}{1-\gamma}$$

so

$$n^* \geq \frac{\log(1-\gamma)}{\log \gamma}$$

□

3 Value of Greedy Policy

Actually, it can be reduced as following

Theorem 3.1. *If $V^k \geq V^* - \lambda$, then $V_g \geq V^* - \frac{2\lambda\gamma}{1-\gamma}$.*

due to the definition, we also have $\|Q^k - Q^*\| \leq \lambda\gamma$

Proof. Let the greedy policy be π and the optimal policy be π^*

$$\begin{aligned} V^*(s) - V_g(s) &= V^*(s) - Q^*(s, \pi(s)) + Q^*(s, \pi(s)) - V_g(s) \\ &\leq V^*(s) - Q^*(s, \pi(s)) + Q^*(s, \pi(s)) - Q_g(s, \pi(s)) \\ &= V^*(s) - Q^*(s, \pi(s)) + \gamma \mathbb{E}_{s'}[V^*(s') - V_g(s')] \end{aligned}$$

$$\begin{aligned} V^*(s) - Q^*(s, \pi(s)) &= V^*(s) - Q_k(s, \pi(s)) + Q_k(s, \pi(s)) - Q^*(s, \pi(s)) \\ &\leq \lambda\gamma + V^*(s) - Q_k(s, \pi^*(s)) \\ &= \lambda\gamma + Q^*(s, \pi^*(s)) - Q_k(s, \pi^*(s)) \\ &\leq 2\lambda\gamma \end{aligned}$$

$$V^*(s) - V_g(s) \leq 2\lambda\gamma + \gamma \mathbb{E}_{s'}[V^*(s') - V_g(s')]$$

Consider the infinity case, we have $V_g(s) \geq V^*(s) - \frac{2\lambda\gamma}{1-\gamma}$ □

4 Written

It is harder to converge for stochastic environment, which means the number of iterations will grow. The optimal policy will also change. Specifically, In policy iteration, the number of steps may stay, but the optimal policy changes; in value iteration, the number grows a lot (7 to 27 in my implementation). More results can be found when running my code.