Deep Reinforcement Learning Assignment 1

EnHsien Chou
2018011438, Department of Computer Science and Technology

1 Optimal Policy for Simple MDP

My answer for (b)(c)(d) is based on the given MDP in Figure 1

(a) If $\gamma > 0$, Let π^* be the optimal policy, $\pi^*(s_i) = a_0$, then ($\forall\ 1 \leq i \leq n-1$):

$$V_{\pi^*}(G) = r_G + \gamma r_G + ... = \frac{1}{1-\gamma}$$

$$V_{\pi^*}(s_{n-1}) = \gamma r_G + \gamma^2 r_G + ... = \frac{\gamma}{1-\gamma}, ...$$

$$\Rightarrow V_{\pi^*}(s_i) = \frac{\gamma^{n-i}}{1-\gamma}$$

If $\gamma = 0$, then $V_{\pi^*}(G) = 1, V_{\pi^*}(s_i) = 0$

(b) No. No matter how the discount factor γ changes from (0,1), $\pi^*(s_i) = a_0$ is still the only optimal policy, it only affects the value of the value function; however, if $\gamma = 0$, $\pi^*(s_i) = a_0$ is still an optimal policy, but there are multiple optimal policies.

(c) No. Let V_π be the old value function and V'_π be the new value function after changing the rewared. When every $r'_t \leftarrow r_t + c$, The optimal policy is still the same: $\pi^*(s_i) = a_0$, but the value of the value function for every policy π changes:

$$V'_\pi(s_i) = \Sigma_{t=0}^{\infty}\gamma^t r'_t = \Sigma_{t=0}^{\infty}\gamma^t r_t + \Sigma_{t=0}^{\infty}\gamma^t c = V_\pi(s_i) + \frac{c}{1-\gamma}$$

(d) Yes. When every $r'_t \leftarrow \alpha(r_t + c)$

$$V'_\pi(s_i) = \Sigma_{t=0}^{\infty}\gamma^t r'_t = \alpha(V_\pi(s_i) + \frac{c}{1-\gamma})$$

Since $V_\pi(s_i) \geq 0$, It depends on the sign of α :

- If $\alpha > 0$, the optimal policy is still the same: $\pi^*(s_i) = a_0$
- If $\alpha = 0$, all states have reward 0, so every policy is an optimal policy with $V' = 0$
- If $\alpha < 0$, every policy satisfying $V_\pi(s_i) = 0$ is an optimal policy, which is any policy that would never reach the goal state

2 Running Time of Value Iteration

(a) $V_1 = 0 + \gamma + \gamma^2 + ... = \frac{\gamma}{1-\gamma}$

(b) $V_2 = \frac{\gamma^2}{1-\gamma}$, Since $V_1 > V_2$, the optimal action is a_1.

(c) For the given MDP, the value iteration updates can be written into:

$$\hat{V}_i(s_0, a_2) = \frac{\gamma^2}{1-\gamma}, \forall i \geq 0$$

$$\hat{V}_{n+1}(s_0, a_1) = R(s_0, a_1) + \gamma V_n(s_1) = \gamma V_n(s_1)$$

Since $V_{n+1}(s_1) = 1 + \gamma V_n(s_1)$

$\Rightarrow \hat{V}_{n+1}(s_0, a_1) = \gamma(1 + \gamma V_{n-1}(s_1)) = \gamma(1 + \gamma + ... + \gamma^{n-1} + \gamma^n V_0(s_1)) = \frac{\gamma(1-\gamma^n)}{1-\gamma}$

Value iteration continues to choose the sub-optimal action (a_2) until:

$$\frac{\gamma(1-\gamma^{n^*})}{1-\gamma} = \hat{V}_{n^*}(s_0, a_1) \geq \hat{V}_{n^*}(s_0, a_2) = \frac{\gamma^2}{1-\gamma}$$

$$\Rightarrow \gamma^{n^*} \leq 1 - \gamma$$

By taking log, notice that $\gamma < 1$ so the sign will change:

$$\Rightarrow n^* \geq log_\gamma(1-\gamma) = \frac{log(1-\gamma)}{log\gamma} \geq \frac{1}{2} log(\frac{1}{1-\gamma}) \frac{1}{1-\gamma}$$

3 Value of Greedy Policy

Proof: $\|V^k - V^*\| \leq \lambda \Rightarrow \|V_g - V^*\| \leq \frac{2\lambda\gamma}{1-\gamma}$

Sol: For a greedy policy $\pi_{\hat{V}}$ derived from $\hat{V}$, we define the loss function:

$$L_{\hat{V}}(x) = V^*(x) - V_{\hat{V}}(x)$$

Assume z is the state which achieves maximum loss, then $L_{\hat{V}}(z) \geq L_{\hat{V}}(x)$. We consider an optimal action $a = \pi^*(z)$ for state z. We consider the action mapped by the policy : $b = \pi_{\hat{V}}(z)$. Since $\pi_{\hat{V}}$ is a greedy policy, we have:

$$R(z,a) + \gamma \sum p(z|y,a)\hat{V}(y) \leq R(z,b) + \gamma \sum p(z|y,b)\hat{V}(y)$$

since for all y, $V^*(y) - \lambda \leq \hat{V}(y) \leq V^*(y) + \lambda$

$$\Rightarrow R(z,a) + \gamma \sum p(z|y,a)(V^*(y) - \lambda) \leq R(z,b) + \gamma \sum p(z|y,b)(V^*(y) + \lambda)$$

$$\Rightarrow R(z,a) - R(z,b) \leq 2\gamma\lambda + \gamma \sum (p(z|y,b)V^*(y) - p(z|y,a)V^*(y))$$

The maximum loss of state z:

$$L_{\hat{V}}(z) = V^*(z) - V_{\hat{V}}(z) = R(z,a) - R(z,b) + \gamma \sum (p(z|y,a)V^*(y) - p(z|y,b)V^*(y))$$

$$\leq 2\gamma\lambda + \gamma \sum (p(z|y,b)V^*(y) - p(z|y,a)V^*(y)) + \gamma \sum (p(z|y,a)V^*(y) - p(z|y,b)V^*(y))$$

$$\Rightarrow L_{\hat{V}}(z) \leq 2\gamma\lambda + \gamma \sum p(z|y,b)[V^*(y) - V_{\hat{V}}(y)]$$

$$\Rightarrow L_{\hat{V}}(z) \leq 2\gamma\lambda + \gamma \sum p(z|y,b)L_{\hat{V}}(y)$$

Since we assume z is the state with max loss, we have $L_{\hat{V}}(z) \geq L_{\hat{V}}(y)$

$$\Rightarrow L_{\hat{V}}(z) \leq 2\gamma\lambda + \gamma \sum p(z|y,b)L_{\hat{V}}(y) \leq 2\gamma\lambda + \gamma \sum p(z|y,b)L_{\hat{V}}(z)$$

Notice that $\sum p(z|y,b) \leq 1$

$$\Rightarrow L_{\hat{V}}(z) \leq 2\gamma\lambda + \gamma L_{\hat{V}}(z)$$

$$\Rightarrow V^*(z) - V_{\hat{V}}(z) = L_{\hat{V}}(z) \leq \frac{2\gamma\lambda}{1-\gamma}$$

Since z is the state with max loss, we can sure that the loss of all states are bounded by $\frac{2\gamma\lambda}{1-\gamma}$. Here finishes the proof.

4 Frozen Lake MDP

(a) See the implemented code in "vi_ and _ pi.py"

(b) See the implemented code in "vi _ and_ pi.py"

(c) The result is shown in the below figure. For value iteration, the stochastic environment needs more iterations to converge. For policy iteration, the number of iterations is unfixed (depending on how we implement). In this problem, the optimal policies for the deterministic and the stochastic are different.

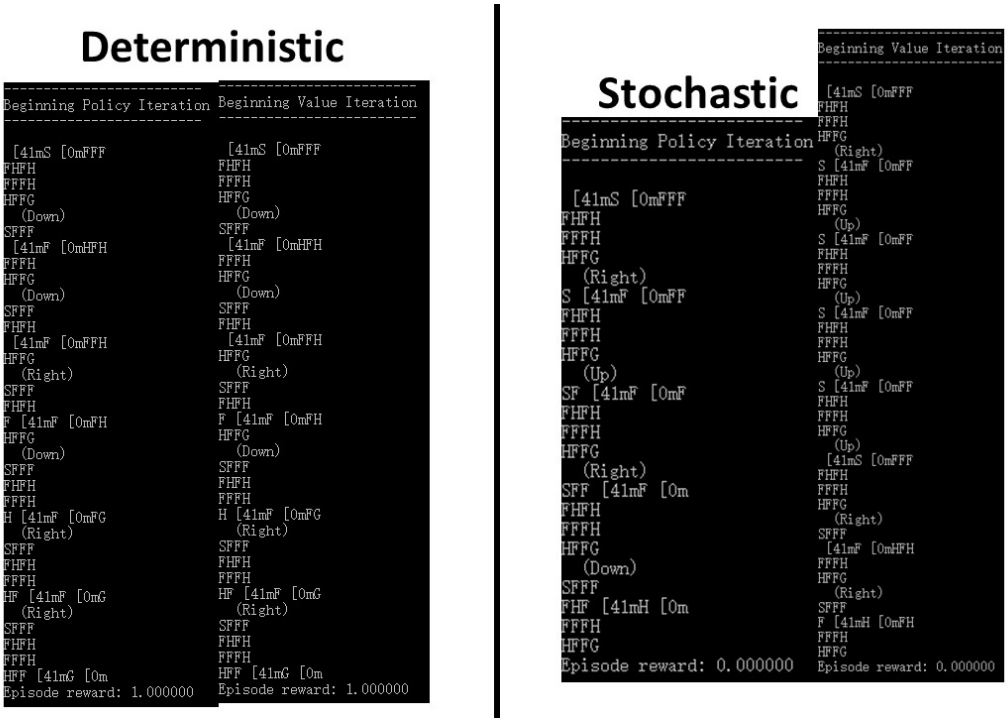

Figure 1: Result for the Frozen Lake MDP

5 Q-Learning Efficiency for Finite-Horizon Problems

The essay shows an efficient reinforcement learning algorithm with posteria sampling exploration. Similar to some concepts of Bayesian learning, we can use posterior probability to optimize the process of exploring. Take Figure 2 for example:

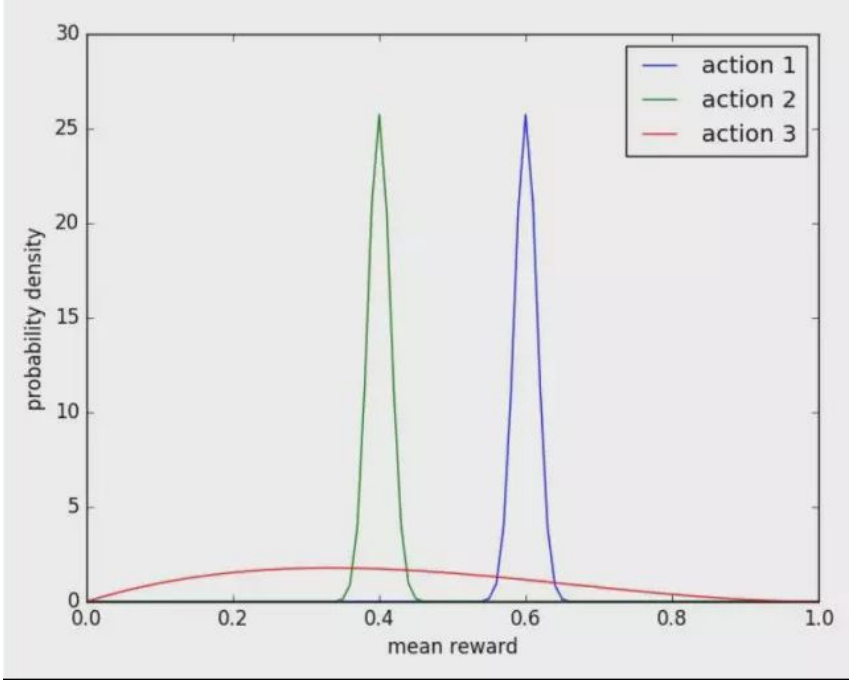

Figure 2: Example of posteria sampling exploration

When using epsilon-greedy exploration, since the algorithm is uniformly selected, the agent will also explore action 2 many times. However, from the graph we see that the reward of action 2 is always less than action 1. Therefore, it is very "expensive" to explore action 2 since there is always a better action 1.

And if we use posteria sampling, it will change the posterior probability for the state(similar to Bayesian learning), so the agent would hold different confidence toward different actions and explore more efficiently.

There are many RL algorithms using posteria sampling exploration, such as Thompson Sampling and Bootstrap DQN, etc.

(The mathematical derivation and the algorithm is too hard for me, therefore I focus more on basic concepts and my opinions)