RL: HW 1

Raymond Koopmanschap

September 20, 2019

2.2 Homework Dynamic Programming

1.

$$\text{Stochastic: } V^\pi(s) = \sum_a \pi(s|a) q^\pi(s,a)$$
$$\text{Deterministic: } V^\pi(s) = \max_a q^\pi(s,a)$$

2. See 3

3. Below the full algorithm, this is for question 2 and 3

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $q \leftarrow Q(s, \pi(s))$
   $Q(s, \pi(s)) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + Q(s', \pi(s'))]$
   $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
   until $\Delta < \theta$
3. Policy Improvement
   $policy - stable \leftarrow true$
   For each $s \in \mathcal{S}$:
   $old - action \leftarrow \pi(s)$
   $\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r|s, a)[r + Q(s', \pi(s'))]$
   If $old - action \neq \pi(s)$, then $policy - stable \leftarrow false$
   If policy-stable, then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to 2

4. Q-value iteration

$$q_{k+1}(s,a) \doteq \sum_{s',r} p(s',r|s,a)[r + \max_{a'} Q(s',a')]$$

3.1 Homework: Monte Carlo

1. (a)

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t|S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right] \\ &= \frac{1}{n}\sum_n G_n = \frac{1}{3}(5 * 0.9^2 + 5 * 0.9^4 + 5 * 0.9^3) = 3.6585 \end{aligned}$$

(b)

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t|S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right] \\ &= \frac{1}{n}\sum_n G_n = \frac{1}{3}(5 + 5 + 5) = 5 \end{aligned}$$

2. Ordinary importance sampling has very high variance. For example if we get a reward of 5 in the behavior policy for a given episode, but in the greedy policy the likelihood of the same episode is 10 times higher, we will then update with a reward of 50.

3. Weighted importance sampling is bias. You always get an importance weight of 1, which is not right since the behavior policy and greedy policy have different distributions.

4.4 Maximization Bias

1. The SARSA and Q-learning Q-values. Hereby assuming we follow a random policy and a decaying learning rate, with each action the same probability.

	SARSA	Q-learning
Q(A,right)	1.5	1.5
Q(A,left)	1	2
Q(B,a_0)	1	1
Q(B,a_1)	1	1
Q(B,a_2)	2	2
Q(B,a_3)	0	0

2. SARSA update state B with samples from a_1, a_2, a_3, a_4, and the expected value of those samples together is 1 (with random policy). However Q-learning update with the max of a_1, a_2, a_3, a_4 and that is 2, so this makes Q(A, left) converge to 2 because every time the same sample is seen in this case. Q-learning has a maximization bias because it is more likely to select overestimated values then underestimated values. Then the same value is used to evaluate that action, so it will look good.

3. Now consider the case that we continue to sample episodes, so we sample again the rewards of a_1, a_2, a_3, a_4 from the Uniform(0, 2) distribution. In this case for example a_2 has a sample of 2,2 and we will update Q_1, thus the argmax for Q_1 is a_2, however instead of updating with the Q-value of this action, we update with $Q_2(B, a_2)$ and this sample is also randomly drawn and thus the expected value is 1.
   Q(A,left) will thus become 1 after convergence.

4. True state-action values

	SARSA	Q-learning	double-Q
Q(A,right)	1.5	1.5	1.5
Q(A,left)	1	1	1
Q(B,a_0)	1	1	1
Q(B,a_1)	1	1	1
Q(B,a_2)	1	1	1
Q(B,a_3)	1	1	1

   Even the Q-learning after a while learns the correct action-value. Since for example given a greedy policy and an ϵ-greedy behavior policy, it will sample the action with the most reward more often and learns that it is on average 1, then it will pick another action more often, until it has learned for all actions that it has on average an expected value of 1.