AI 3000 / CS5500 : Reinforcement Learning
Exam № 2

Course Instructor : Easwar Subramanian 25/11/2023, 9.00 AM

Problem 1 : Q Learning

Consider a single state state MDP with two actions. That is, $\mathcal{S} = \{s\}$ and $\mathcal{A} = \{a_1, a_2\}$. Assume the discount factor of the MDP is γ and the horizon length to be 1. Both actions yield random rewards, with expected reward for each action being a constant $c \geq 0$. That is,

$$\mathbb{E}(r|a_1) = c \text{ and } \mathbb{E}(r|a_2) = c$$

where $r \sim \mathcal{R}^{a_i}$, $i \in \{1, 2\}$.

(a) What are the true values of $Q(s, a_1)$, $Q(s, a_2)$ and $V^*(s)$? (1 Point)

$Q(s, a_1) - Q(s, a_2) = V(s) = c$

(b) Consider a collection of n prior samples of reward r obtained by choosing action a_1 or a_2 from state s. Denote $\hat{Q}(s, a_1)$ and $\hat{Q}(s, a_2)$ to be the sample estimates of action value functions $Q(s, a_1)$ and $Q(s, a_2)$, repectively. Let $\hat{\pi}$ be a greedy policy obtained with respect to the estimated $\hat{Q}(s, a_i)$, $i \in \{1, 2\}$. That is,

$$\hat{\pi}(s) = \arg\max_a \hat{Q}(s, a)$$

Prove that the estimated value of the policy $\hat{\pi}$, denoted by $\hat{V}^{\hat{\pi}}$, is a biased estimate of the optimal value function $V^*(s)$. (4 Points)

[**Note** : Assume that actions a_1 and a_2 have been choosen equal number of times.]

The unbiasaed sample estimates for $Q(s, a_i) = \frac{1}{n}\sum_{ij=1}^{n} r_j$ for $i = \{1, 2\}$ $\hat{\pi}$ is the greedy policy with respect to $\hat{Q}$ Then

$$\begin{aligned} \hat{V}^{\hat{\pi}}(s) &= \mathbb{E}(\max(\hat{Q}(s, a_1), \hat{Q}(s, a_2)) \\ &\geq \max(\mathbb{E}(\hat{Q}(s, a_1), \hat{Q}(s, a_2)) \\ &= \max(c, c) = V^{\pi}(s) \end{aligned} \tag{1}$$

The second inequivality is due to Jensen's inequality.

(c) Let us now consider that the first action a_1 always gives a constant reward of c whereas the second action a_2 gives a reward $c + \mathcal{N}(-0.1, 1)$ (normal distribution with mean -0.1 and unit variance). Which is the better action to take in expectation ? Would the TD control algorithms like Q-learning or SARSA control, trained using finite samples, always favor the action that is best in expectation ? Explain. (2 Points)

Action a_1 is a better action in expectation. But Q-learning and SARSA control can choose action a_2 because they use sample estimates to decide the best action

(d) Suggest with justification another update rule in lieu of Q-learning or SARSA control that is likely to favor the better action for the previous sub-problem. (3 Points)

Double Q learning. Provide update rule for double q learning. Double q learning is for maximization bias. Double q learning update rule solves it by having separate Q function for chosing the best action and q-table update. For each transition quadruple $(s_t, a_t, r_{t+1}, s_{t+1})$ we flip a fair coin to decide any of the two update steps given below,

$$Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q_2(s_{t+1}, \arg\max Q_1(s_{t+1}, a)) - Q_1(s_t, a_t)\right)$$

$$Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q_1(s_{t+1}, \arg\max Q_2(s_{t+1}, a)) - Q_2(s_t, a_t)\right)$$

Note : Q-learning update formula :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

Problem 2 : Policy Gradients

1. Define advantage function $A^\pi(s, a)$. What does it aim to capture ? For a given policy π, what is $\mathbb{E}_{a\sim\pi}(A^\pi(s, a))$? (3 Points)

   - Definition :
   $$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$
   - Advantage function estimates the edge that a particular action a has over all possible actions of policy π from state s.
   - $\mathbb{E}_{a\sim\pi}(A^\pi(s, a)) = 0$ as expectation of Q is V

2. How is advantage funtion computed in an unbiased fashion ? (2 Points)
   Consider one-step TD error for V^{π_θ}

   $$\delta_t^{\pi_\theta} = r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$$

   $$\begin{aligned} \mathbb{E}_{\pi_\theta}(\delta^{\pi_\theta}|s_t, a_t) &= \underbrace{E_{\pi_\theta}(r_{t+1} + \gamma V^{\pi_\theta}(s_{t+1})|s_t, a_t)}_{??} - V^{\pi_\theta}(s_t) \\ &= Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) = A^{\pi_\theta}(s_t, a_t) \end{aligned}$$

3. Would MC based policy gradient methods converge faster or slower compared to their actor-critic counterparts assumping similar parameterizations to policies ? If faster, explain why ? If slower, suggest ways to improve the speed of convergence (2 Points)

   MC based methods converge slowly as compared to their actor-critic counterparts as their variance is on high. Variance of MC methods can be reduced by the use of optimal baselines, temporal structure, discounting. Once could also have parallel environments, which

should be uncorrelated (i.e., initialized all differently), allowing you to collect a batch of trajectories that you can average on to reduce a bit the noise due to the stochastic nature of the agent-environment interaction.

4. Policy gradient algorithms aim to find optimal (stochastic) policies for a given MDP. What limitations of the vanilla policy gradient algoritahms does the natural gradient based techniques overcome ? How does the formulation of the optimization problem different in the two settings ? What difficulties does a practitioner need to face while implementing natural gradient based methods ? (5 Points)

   - Limiatations : Sample inefficiency and Distance in parameter space $\neq$ Distance in policy space
   - Formulation is dfiferent in terms on constraint (should give the formulation) where we search for policies within a KL distance of δ
   - Computing KL estimates using samples, inverating FIM, approximations lin formulations leading to other problems such as lack of monotonic improvement of successive policies.

Problem 3 : Bandit Algorithms

1. Which of the two algorithms among ε**-greedy** and ε**-greedy with** ε**-decay** have better total regret ? Why ? (2 Points)

   ε**-greedy with** ε**-decay** with a GLIE type of decay scheduling would have better regret (sublinear) as such algorithms neither explore or exploit forever

2. Consider a multi-armed bandit with two arms. Arm 1 always gives a reward of 1 on each pull. Arm 2 always gives a reward of 0 on each pull. Other than the initialization phase, when each arm is pulled once, would arm 2 ever get pulled again? If no, explain why. If yes, provide, with justifcation, a condition that needs to be satisfied for arm 2 to be pulled again. (3 Points)

   Yes, Arm 2 will be pulled when the UCB term of Arm 2 becomes greater than arm 1. If there have been a total of N pulls with 1 for arm 2 and N-1 for arm 1 then the UCB co-efficients of Arm 2 and Arm 1 are : $\sqrt{\log N}$ and $1 + \sqrt{\frac{\log N}{N-1}}$ respectively.

3. Consider a multi-armed bandit with five arms. All five arms are identical and give a reward of 1 on each pull. After 10 arm pulls using the UCB algorithm will each arm would have been pulled twice, regardless of how one break ties when arms have equal UCB values ? Justify your answer. (3 Points)

   Yes, all arm will be pulled twice irrespective of how tie breaks are done. After the initialization phase, all arm would have been pulled once and their UCB values would be tied. The algorithm does not care how we break ties, it will give the same result for the sixth

arm pull regardless of which of the arms we pull. Let's arbitrarily pull arm 1. Thereafter, we will have a tie between arms 2-5 and their UCB value will be lower than arms 2-5. If we arbitrarily choose to pull arm 2 from those 4 arms and we keep doing this, by the end of the 10 pulls, we would have pulled each arm twice.

Note : For sub-problems (b) and (c) use the following UCB formula :

$$a_t = \arg\max_a \left[\underbrace{\hat{Q}_t(a)}_{\textbf{Exploitation}} + \underbrace{\sqrt{\frac{\log N}{N_t(a)}}}_{\textbf{Exploration}} \right]$$

ALL THE BEST