Indian Institute of Technology Kharagpur
Department of Computer Science and Engineering

Reinforcement Learning (CS60077) **Autumn Semester, 2022-2023**

End-Semester Examination **25-Nov-2022 (Friday), 14:00–17:00** **Maximum Marks: 100**

Instructions:

- Write your answers in the answer booklet provided to you in the examination hall.
- There are a total of SIX questions. Marks are indicated in parentheses. All questions are compulsory. Write the answers for all parts of a question together.
- Be brief and precise. Organize your work, in a reasonably neat and coherent way. Work scattered all across the answer script without a clear ordering will receive very little marks.
- If you use any theorem / result / formula covered in the class, just mention it and do not elaborate and/or prove (if not explicitly asked to do so).
- Write all the derivations / proofs / deductions in mathematically and/or logically precise language. Unclear and/or dubious statements would be severely penalized.
- Mysterious or unsupported answers will not receive full marks. A correct answer, unsupported by calculations, explanation, will receive no marks; an incorrect answer supported by substantially correct calculations and explanations may receive partial marks.

– The question paper starts from the next page –

Q1. [30 Marks] Choose all the correct answer(s) with a brief justification for each of them. No marks will be awarded without justification or partially made choices (in case of multiple correct answers).

(a) Suppose we want an agent to learn to play the game of golf. For training purposes, we make use of a golf simulator program. Assume that the original reward distribution gives a reward of $+10$ when the golf ball is hit into the hole and -1 for all other transitions. To aid the agents learning process, we propose to give an additional reward of $+3$ whenever the ball is within a 1-metre radius of the hole. Is this additional reward a good idea or not? Why? **(3)**

(i) No, the additional reward may actually hinder learning
(ii) No, it violates the idea that a goal must be outside the agents direct control
(iii) Yes, the additional reward will help speed-up learning
(iv) Yes, getting the ball to within a metre of the hole is like a sub-goal and hence, should be rewarded

(b) Consider a 100×100 grid world domain where the agent starts each episode in the bottom-left corner, and the goal is to reach the top-right corner in the least number of steps. To learn an optimal policy to solve this problem, you decide on a reward formulation in which the agent receives a reward of $+1$ on reaching the goal state and 0 for all other transitions. Suppose you try two variants of this reward formulation: **(P1)**, where you use discounted returns with $\gamma \in (0, 1)$, and **(P2)**, where no discounting is used (i.e., $\gamma = 1$). Which among the following would you expect to observe? **(3)**

(i) The same policy is learned in (P1) and (P2)
(ii) No policy learning in (P1)
(iii) No policy learning in (P2)
(iv) Policy learned in (P2) is better than the policy learned in (P1)

(c) Given a policy π over a Markov Decision Process (MDP) having state value function $v_\pi(s)$ and action value function $q_\pi(s, a)$ over state s with action a, if $q_\pi(s, a) > v_\pi(s)$, which of the following can be concluded? **(3)**

(i) π is not an optimal policy
(ii) π may be an optimal policy
(iii) Action a is the best action that can be taken in state s
(iv) None of the above

(d) In the iterative policy evaluation process, we have seen the use of different update equations in Dynamic Programming (DP), Monte-Carlo (MC), and Temporal Difference (TD) methods. With regard to these update equations, which of the following statements is/are correct? **(3)**

(i) DP and TD make use of estimates but not MC
(ii) TD makes use of estimates but not DP and MC
(iii) MC and TD make use of estimates but not DP
(iv) All three methods make use of estimates

(e) Which among the following statements about LSTD and LSTDQ methods is/are *incorrect*? **(3)**

(i) LSTD learns the state value function
(ii) LSTDQ learns the action value function
(iii) Both LSTD and LSTDQ can reuse samples
(iv) Both LSTD and LSTDQ can be used along with tabular representations of value functions

(f) Suppose we are using a policy gradient method to solve a reinforcement learning problem. Assuming that the policy returned by the method is not optimal, which among the following is/are *not* the plausible reason(s) for such an outcome? **(3)**

(i) Search procedure converged to a locally optimal policy
(ii) Search procedure was terminated before it could reach the optimal policy
(iii) Sample trajectories arising in the problem were very long
(iv) Optimal policy could not be represented by the parameterization used to represent the policy

(g) For a given policy π, if f_w approximates Q_π and is compatible with the parameterization used for the policy, then this indicates that we can use f_w in place of Q_π in the expression for calculating the gradient of the policy performance metric with respect to the policy parameter. Which among the following is/are appropriate reasons for this? **(3)**

(i) $Q_\pi(s,a) - f_w(s,a) = 0$ in the direction of the gradient of $f_w(s,a)$
(ii) $Q_\pi(s,a) - f_w(s,a) = 0$ in the direction of the gradient of $\pi(s,a)$
(iii) The error between Q_π and f_w is orthogonal to the gradient of the policy parameterization
(iv) The reason is not listed above

(h) After 12 iterations of the UCB 1 algorithm applied on a 4-arm bandit problem, we have $n_1 = 3,\ n_2 = 4,\ n_3 = 3,\ n_4 = 2$ and $Q_{12}(1) = 0.55,\ Q_{12}(2) = 0.63,\ Q_{12}(3) = 0.61,\ Q_{12}(4) = 0.40$. Which arm should be played next? **(3)**

(i) 1
(ii) 2
(iii) 3
(iv) 4

(i) Consider the following problem design. You have a grid world with several rooms, as discussed in the lectures, with the goal state in a corner cell of one of the rooms. You set up an agent with options for exiting each of the rooms into the other. You also allow the agent to pick from the four primitive actions. There is a step reward of -1. The learning algorithm used is SMDP Q-learning, with normal Q-learning updates for the primitive actions. You expect the agent to learn faster due to the presence of the options, but discover that it is not the case. Can you explain what might have caused this? **(3)**

(i) The options are not useful for solving the problem, hence caused the slowdown
(ii) Initially, the agent will focus more on using primitive actions, causing the slowdown
(iii) Due to the presence of options, we can no longer achieve the optimal solution, hence it takes longer duration
(iv) It takes longer duration because we are using SMDP Q-learning compared to conventional Q-learning which is faster

(j) Solving an hierarchical reinforment learning problem using the MAXQ approach leads to which of the following type of solutions? **(3)**

(i) Hierarchically optimal
(ii) Recursively optimal
(iii) Flat optimal
(iv) Non-optimal

Solution:

(a) Correct Choice: **(i)**

In this specific case, the additional reward will be detrimental to the learning process, as the agent will learn to accumulate rewards by keeping the ball within the 1 metre radius circle and not actually hitting the ball in the hole. This example highlights the importance of being cautious in setting up the reward function in order to prevent unintended consequences.

(b) Correct Choice: **(iii)**

In (P2), since there is no discounting, the return for each episode regardless of the number of steps is $+1$. This prevents the agent from learning a policy which tries to minimise the number of steps to reach the goal state. In (P1), the discount factor ensures that the longer the agent takes to reach the goal state, the lesser reward it gets. This motivates the agent to find take the shortest path to the goal state.

(c) Correct Choice: **(i)**

The inequality indicates that there exists an action that if taken in state s, the expected return would be higher than the expected return of taking actions in state s as per policy π. While this indicates that π is not an optimal policy, it does not indicate that a is the best action that can be taken in state s, since there may exist another action a' such that $q_\pi(s, a') > q_\pi(s, a)$.

(d) Correct Choice : **(iv)**

DP methods update estimates based on other learned estimates, i.e., they bootstrap. MC methods, while they do not bootstrap, make use of estimates because the target in MC methods, i.e., the sample return, is an estimate of the actual expected return; the expected value of course, is not known. TD methods use both the above, i.e., they make use of samples of the expected values as well as bootstrap.

(e) Correct Choice: **(iii)**

Option (iii) is not correct. Given a set of samples collected from the actual process (not from a generative model in which case reusing samples is perhaps not that important) it is useful for these samples to be reused in evaluating different policies. Recall that LSPI is a policy iteration algorithm where the policy is constantly being improved upon (and hence changing). Such sample reuse is possible in LSTDQ if for any policy π, $\pi(s')$ is available for each s' in the set of samples. This is because for different policies, the same samples can be made use of, as individual policies only determine the $\phi(s', \pi(s'))$ component of $\hat{A}$. This is not the case with LSTD where freedom in action choices is not available and must be determined from the policy that is being evaluated.

(f) Correct Choice: **(iii)**

Option (iii) may result in an increase in the time it takes to converge to a policy, but does not necessarily affect the optimality of the policy obtained.

(g) Correct Choices: **(ii), (iii)**

As indicated by options (ii) & (iii), we can use f_w in place of Q_π if the difference between the two is zero in the direction of the gradient of $\pi(s, a)$.

(h) Correct Choice: **(iv)**

The next action, A_{13}, will be the action with the maximum upper confidence bound among the four arms. Calculating these values, we have

$$Q_{12}(1) + \sqrt{\frac{2\ln 12}{n_1}} = 0.55 + \sqrt{\frac{2\ln 12}{3}} = 1.837 \qquad Q_{12}(2) + \sqrt{\frac{2\ln 12}{n_2}} = 0.63 + \sqrt{\frac{2\ln 12}{4}} = 1.745$$
$$Q_{12}(3) + \sqrt{\frac{2\ln 12}{n_3}} = 0.61 + \sqrt{\frac{2\ln 12}{3}} = 1.897 \qquad Q_{12}(4) + \sqrt{\frac{2\ln 12}{n_4}} = 0.40 + \sqrt{\frac{2\ln 12}{2}} = 1.976$$

Clearly, arm 4 has the highest upper confidence bound and hence will be selected by the UCB 1 algorithm.

(i) Correct Choice: **(ii)**

Consider the contrast between selecting a primitive action and an option during the initial phase of learning in this problem. On selecting a primitive action from states near the start state or a doorway state, you do not reach the goal state and get a unit of negative reward. On selecting an option, however, you are transported to some doorway state (which is not the goal state) and get a negative reward of larger magnitude (recall SMDP rewards). This suggests that initially, the primitive options will appear more promising and will be explored more whereas the benefit of the options will not initially be realised or exploited. Thus, the expected speedup would in general not be observed.

(j) Correct Choice: **(ii)**

Since the MAXQ policy of the core MDP is the set of policies of individual sub-tasks, with individual sub-task policies aiming to solve the sub-tasks optimally, you can expect to obtain recursively optimal solutions using the MAXQ approach.

Q2. [16 Marks] Let $M = (S, A, R, P, \gamma)$ be an MDP with $|S| < \infty$, $|A| < \infty$ and $\gamma \in [0, 1)$. Let $\hat{M}$ be a modification of M to be specified below. We shall compare the optimal value functions and policies in M and $\hat{M}$. Assume that, V^* and $\hat{V}^*$ are the optimal value functions in M and $\hat{M}$, respectively.

(a) Let $\hat{M} = (S, A, \hat{R}, P, \gamma)$ where $|\hat{R}(s, a) - R(s, a)| \leq \epsilon$ for all $s \in S$ and $a \in A$. Besides the rewards, all other components of $\hat{M}$ stay the same as in M.

(i) Prove that, $V^* - \hat{V}^* \leq \frac{\epsilon}{1-\gamma}$. **(6)**

(ii) Will M and $\hat{M}$ have the same optimal policy? Briefly explain. **(2)**

(b) Now, let $\hat{M} = (S, A, \hat{R}, P, \gamma)$ where $\hat{R}(s, a) - R(s, a) = \epsilon$ for all $s \in S$ and $a \in A$ for some constant ϵ. Equivalently, the same constant ϵ is added to each reward $R(s, a)$ to obtain $\hat{R}(s, a)$.

(i) How are V^* and $\hat{V}^*$ related? Express $\hat{V}^*$ in terms of V^*. **(5)**

(ii) Will M and $\hat{M}$ has the same optimal policy? Briefly explain. **(3)**

Solution:

(a) (i) Define the state $\tilde{s}$ as follows: $\tilde{s} = arg \max_s \big[V^*(s) - \hat{V}^*(s)\big]$

Also, the inequality $|R(s, a) - \hat{R}(s, a)| \leq \epsilon$ implies: $R(s, a) \leq \hat{R}(s, a) + \epsilon$, for all (s, a).

As a result:

$$\begin{aligned}
V^*(\tilde{s}) &= \max_{a\in A} \Big[R(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] V^*(s')\Big] \\
&\leq \max_{a\in A} \Big[\hat{R}(s,a) + \epsilon + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \big\{V^*(s') - \hat{V}^*(s') + \hat{V}^*(s')\big\}\Big] \\
&\leq \max_{a\in A} \Big[\hat{R}(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \hat{V}^*(s')\Big] + \max_{a\in A} \Big[\gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \big(V^*(s') - \hat{V}^*(s')\big)\Big] + \epsilon \\
&\leq \hat{V}^*(\tilde{s}) + \gamma\big[V^*(\tilde{s}) - \hat{V}^*(\tilde{s})\big] + \epsilon
\end{aligned}$$

Therefore, $V^*(\tilde{s}) \leq \hat{V}^*(\tilde{s}) + \gamma\big[V^*(\tilde{s}) - \hat{V}^*(\tilde{s})\big] + \epsilon$, implies $V^*(\tilde{s}) - \hat{V}^*(\tilde{s}) \leq \frac{\epsilon}{1-\gamma}$.

Hence, for all states $s \in S$, we can say that $V^*(s) - \hat{V}^*(s) \leq \frac{\epsilon}{1-\gamma}$.

(ii) $V^*(s)$ and $\hat{V}^*(s)$ do not necessarily have the same optimal policy. This is because the difference in rewards does not necessarily preserve that ordering amongst actions in the state-action values.

(b) (i) For all s, $V^*(s) = \max_{a\in A} \Big[R(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] V^*(s')\Big]$

Since $\frac{\epsilon}{1-\gamma} = \epsilon + \gamma\frac{\epsilon}{1-\gamma}$, therefore

$$\begin{aligned}
V^*(s) + \frac{\epsilon}{1-\gamma} &= \max_{a\in A} \Big[R(s,a) + \epsilon + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \big(V^*(s') + \epsilon\big)\Big] \\
\implies V^*(s) + \frac{\epsilon}{1-\gamma} &= \max_{a\in A} \Big[\hat{R}(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \big(V^*(s') + \epsilon\big)\Big]
\end{aligned}$$

As a result, for all $s \in S$: $\hat{V}^*(s) = V^*(s) + \frac{\epsilon}{1-\gamma}$.

(ii) In addition,

$$arg \max_{a\in A} \Big[\hat{R}(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a)\big] \hat{V}^*(s')\Big] = arg \max_{a\in A} \Big[R(s,a) + \epsilon + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] \big(V^*(s') + \frac{\epsilon}{1-\gamma}\big)\Big]$$

$$arg \max_{a\in A} \Big[R(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a)\big] V^*(s') + \epsilon + \frac{\epsilon}{1-\gamma}\Big] = arg \max_{a\in A} \Big[R(s,a) + \gamma \sum_{s'\in S} \mathbb{P}\big[s' \mid s,a\big] V^*(s')\Big]$$

So, M and $\hat{M}$ will have the same optimal policy.

Q3. [14 Marks] Assume an underlying Markov Decision Process (MDP), $M = (S, A, P, R, \gamma)$, where $s, s' \in S$ are the states of an MDP, $a \in A$ denotes an action in MDP, $r(s,a) = R_s^a$ denotes the reward accumulated when applying action a to state s, $\mathbb{P}\big[s' \mid s, a\big] = P_{ss'}^a$ denotes the transition probability to s' from s upon executing action a, and $\gamma \in (0, 1]$ is the discount factor.

(a) The Bellman equations give us the iterative relations that relate the state value functions and the action value functions. The Bellman expectation equations provide the relation between state value function v_π and action value function q_π for any policy π, while the Bellman optimality equations provide the relation between the optimal state value function v_* and the optimal action value function q_*. The two tables below require you to fill out the relations between the quantities marked in the row and the column headings. **(3 + 3)**

(i) Fill up the missing entries from the table for *Bellman Expectation Equations* as given below.

	v_π	q_π
v_π	$v_\pi(s) = ?$	$v_\pi(s) = \sum\limits_{a \in A} \pi(a\|s) q_\pi(s,a)$
q_π	$q_\pi(s,a) = ?$	$q_\pi(s,a) = ?$

(ii) Fill up the missing entries from the table for *Bellman Optimality Equations* as given below.

	v_*	q_*
v_*	$v_*(s) = ?$	$v_*(s) = \max\limits_{a \in A} q_*(s,a)$
q_*	$q_*(s,a) = ?$	$q_*(s,a) = ?$

Note: If you use notations and symbols different from this/class, explain them.

Hints: (1) Left hand side of the relations are functions of the current states or current states and actions, while the right hand side may be functions of current and future values of these quantities. (2) For your convenience one of the relations is provided in each of the table (They are, of course, the easier ones!).

(b) Let π be an ϵ-greedy policy. Let π' be the ϵ-greedy policy inferred from the action value function q_π (ϵ-greedy Policy Improvement from π to π'), i.e.,

$$\pi'(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|}, & \text{if } a = arg \max\limits_{b \in A} q_\pi(s,b) \\ \frac{\epsilon}{|A|}, & \text{otherwise} \end{cases}$$

Prove that, $\sum\limits_{a \in A} \pi'(a|s).q_\pi(s,a) \geq v_\pi(s)$, for all $s \in S$,

where $v_\pi(s)$ is state value function for policy π. **(8)**

Solution:

(a) (i) The table for *Bellman Expectation Equations* is as follows:

	v_π	q_π
v_π	$v_\pi(s) = \sum\limits_{a \in A} \pi(a\|s) \Big[r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] v_\pi(s')\Big]$	$v_\pi(s) = \sum\limits_{a \in A} \pi(a\|s) q_\pi(s,a)$
q_π	$q_\pi(s,a) = r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] v_\pi(s')$	$q_\pi(s,a) = r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] \Big[\sum\limits_{a' \in A} \pi(a'\|s') q_\pi(s',a')\Big]$

(ii) The table for *Bellman Optimality Equations* is as follows:

	v_*	q_*
v_*	$v_*(s) = \max\limits_{a \in A} \Big[r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] v_*(s')\Big]$	$v_*(s) = \max\limits_{a \in A} q_*(s,a)$
q_*	$q_*(s,a) = r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] v_*(s')$	$q_*(s,a) = r(s,a) + \gamma \sum\limits_{s' \in S} \mathbb{P}\big[s' \mid s, a\big] \max\limits_{a' \in A} q_*(s',a')$

(b) It may be noted that,

$$\sum_{a \in A} \pi'(a|s).q_\pi(s,a) = \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \max_{a \in A} q_\pi(s,a)$$

Now, we make the crucial observation that a max over choices of A is greater than or equal to a weighted average over choices of A. Specifically,

$$\max_{a \in A} q_\pi(s,a) \geq \sum_{a \in A} w_a.q_\pi(s,a),$$

for any choice of weights $w_a \geq 0$, $a \in A$ constrained by $\sum\limits_{a \in A} w_a = 1$. We will make a specific choice of w_a as follows:

$$w_a = \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon}$$

We note that $w_a \geq 0$ for all $a \in A$ because $\pi(a|s) \geq \frac{\epsilon}{m}$ (since $\pi(a|s)$ is an ϵ-greedy policy). We also note that,

$$\sum_{a \in A} w_a = \frac{\sum\limits_{a \in A} \pi(a|s) - \sum\limits_{a \in A} \frac{\epsilon}{m}}{1-\epsilon} = \frac{1-\epsilon}{1-\epsilon} = 1$$

Having established that,

$$\max_{a \in A} q_\pi(s,a) \geq \sum_{a \in A} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon}.q_\pi(s,a)$$

we can go back to the initial equation and state that:

$$\frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \max_{a \in A} q_\pi(s,a) \geq \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \sum_{a \in A} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon}.q_\pi(s,a)$$

Therefore,

$$\begin{aligned}
\sum_{a \in A} \pi'(a|s).q_\pi(s,a) \quad &\geq \quad \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + (1-\epsilon) \sum_{a \in A} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon}.q_\pi(s,a) \\
&= \quad \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) + \sum_{a \in A} \pi(a|s).q_\pi(s,a) - \frac{\epsilon}{m} \sum_{a \in A} q_\pi(s,a) \\
&= \quad \sum_{a \in A} \pi(a|s).q_\pi(s,a) \quad = \quad v_\pi(s)
\end{aligned}$$

Hence, $\sum\limits_{a \in A} \pi'(a|s).q_\pi(s,a) \geq v_\pi(s)$, for all $s \in S$. [Proved]

Q4. [14 Marks] Consider an episodic, deterministic chain MDP with $n = 7$ states assembled in a line. The possible actions are, $a \in \{-1, 1\}$, and the transition function is deterministic such that $s' = s + a$, except the fact that, taking $a = -1$ from $s = 1$ keeps us in $s = 1$, and taking $a = 1$ from $s = 7$ keeps us in $s = 7$. We have a special goal state, $s = 4$, such that taking any action from $s = 4$ ends the episode with a reward of $r = 0$. From all other states (i.e., $s \neq 4$ and $1 \leq s \leq 7$), any action incurs a reward of $r = -1$. We let $\gamma = 1$. The chain MDP is provided in Figure 1, with the goal state $s = 4$ marked as *shaded.*

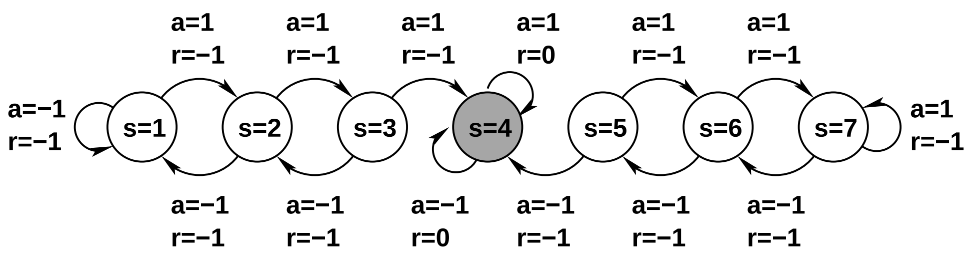


Figure 1: Chain MDP

(a) We wish to perform tabular Q-learning on this chain MDP. First, express clearly the Q-learning state-action value update equation/rule, given a tuple (s, a, r, s'). **(2)**

Suppose we observe the following 4-step trajectory (in the form of $\langle state, action, reward \rangle$) as:

$$\langle 3, -1, -1 \rangle; \ \langle 2, 1, -1 \rangle; \ \langle 3, 1, -1 \rangle; \ \langle 4, 1, 0 \rangle$$

Let all Q values are initialized to 0 (zero). Use the tabular Q-learning update to give updated values for $Q(3, -1)$, $Q(2, 1)$, and $Q(3, 1)$, assuming we process the trajectory in the order given from left to right. Use the learning rate, $\alpha = \frac{1}{2}$. **(3)**

(b) Now, we are interested in performing linear function approximation in conjunction with Q-learning. In particular, we have a weight vector, $w = \big[w_1, w_2, w_3\big]^T \in \mathbb{R}^3$. Given some state s and action $a \in \{-1, 1\}$, the featurization of this state, action pair is: $\phi = \big[s, a, 1\big]^T$. To compute Q-values, we compute $\hat{q}(s, a; w) = w^T \phi = w_0.s + w_1.a + w_2$. Given the parameters w and a single sample (s, a, r, s'), the loss function we will minimize is, $J(w) = \big(r + \gamma \max_{a'} \hat{q}(s', a'; w^-) - \hat{q}(s, a; w)\big)^2$, where $\hat{q}(s', a'; w^-)$ is a target network parametrized by fixed weights w^-.

Suppose, you currently have weight vectors $w = \big[-1, 1, 1\big]^T$ and $w^- = \big[1, -1, -2\big]^T$, and you observe a sample, $(s = 2, a = -1, r = -1, s' = 1)$. Perform a single gradient update to the parameters w given this sample. Use the learning rate $\alpha = \frac{1}{4}$. Write out the gradient $\nabla_w J(w)$ as well as the new parameters w'. Show all deductions clearly. **(7)**

(c) The optimal Q-function $Q^*(s, a)$ is exactly representable by some neural network architecture $\mathcal{N}$. Suppose we perform Q-learning on this MDP using the architecture $\mathcal{N}$ to represent the Q-values. Suppose we randomly initialize the weights of a neural net with architecture $\mathcal{N}$ and collect infinitely many samples using an exploration strategy that is *greedy in the limit of infinite exploration* (GLIE). Are we guaranteed to converge to the optimal Q function $Q^*(s, a)$. Explain your answer. **(2)**

Solution:

(a) Given a tuple (s, a, r, s'), we use the update equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a' \in \{-1,1\}} Q(s',a') - Q(s,a)\Big)$$

Using this equation with $\alpha = \frac{1}{2}$ and $\gamma = 1$, we have:

$$\begin{aligned}
Q(3,-1) &\leftarrow 0 + \frac{1}{2}\Big(-1 + \max_{a'} Q(2,a')\Big) = \frac{1}{2}(-1+0) = -\frac{1}{2} \\
Q(2,1) &\leftarrow 0 + \frac{1}{2}\Big(-1 + \max_{a'} Q(3,a')\Big) = \frac{1}{2}(-1+0) = -\frac{1}{2} \\
Q(3,1) &\leftarrow 0 + \frac{1}{2}\Big(-1 + \max_{a'} Q(4,a')\Big) = \frac{1}{2}(-1+0) = -\frac{1}{2}
\end{aligned}$$

(b) The gradient of $J(w)$ yields,

$$\begin{aligned}
\nabla_w J(w) &= -2\big(r + \gamma \max_{a'} \hat{q}(s',a';w^-) - \hat{q}(s,a;w)\big)\nabla_w \hat{q}(s,a;w) \\
&= -2\big(r + \max_{a'} (w^-)^T[s',a',1]^T - w^T[s,a,1]^T\big)[s,a,1]^T
\end{aligned}$$

Using this, the parameter update with a single sample (s, a, r, s') is:

$$\begin{aligned}
w' &\leftarrow w - \frac{1}{2}\alpha \nabla_w J(w) \\
&= w + \frac{1}{4}\big(r + \max_{a'} (w^-)^T[s',a',1]^T - w^T[s,a,1]^T\big)[s,a,1]^T
\end{aligned}$$

Using the sample $(2,-1,-1,1)$ and the particular values of w and w^- yields:

$$\begin{aligned}
w' &\leftarrow [-1,1,1]^T + \frac{1}{4}\big(-1 + \max_{a'}[1,-1,-2].[1,a',1]^T - [-1,1,1].[2,-1,1]^T\big)[2,-1,1]^T \\
&= [-1,1,1]^T + \frac{1}{4}\big(-1 + \max_{a'}(1 - a' - 2) - (-2-1+1)\big)[2,-1,1]^T \\
&= [-1,1,1]^T + \frac{1}{4}[2,-1,1]^T \quad = \Big[-\frac{1}{2}, \frac{3}{4}, \frac{5}{4}\Big]^T
\end{aligned}$$

<u>Note</u>: Alternatively, the parameter update could also be written as: $w' \leftarrow w - \alpha \nabla_w J(w)$.

This is also fine, and the subsequent answer obtained will be:

$$w' \leftarrow [-1,1,1]^T + \frac{1}{2}[2,-1,1]^T = \Big[0, \frac{1}{2}, \frac{3}{2}\Big]^T$$

(c) Since this method of Q-learning involves *function approximation*, *bootstrapping*, and *off-policy training* (the "deadly triad" according to Sutton & Barto book), we are not guaranteed to converge to anything, which includes no guarantee of converging to the optimal Q-function.

Q5. [13 Marks] Consider a finite MDP with the set of finite state space denoted as S and a set of finite actions space denoted as A. Let $\phi(s,a) = \big[\phi_1(s,a), \phi_2(s,a), \ldots, \phi_n(s,a)\big]$ be the features vector for any $s \in S$, $a \in A$. Let $\theta = \big[\theta_1, \theta_2, \ldots, \theta_n\big]$ be an n-vector of parameters. Let the action probabilities conditional on a given state s and given parameter vector θ be defined by the *softmax function* on the linear combination of features: $\theta^T.\phi(s,a)$, i.e.,

$$\pi(a|s;\theta) = \frac{e^{\theta^T.\phi(s,a)}}{\sum\limits_{b\in A} e^{\theta^T.\phi(s,b)}}$$

(a) Evaluate the score function $\nabla_\theta \log \pi(a|s;\theta)$. Show the complete deduction. **(6)**

(b) Construct the action value function approximation $Q(s,a;w)$ so that the following key constraint of the Compatible Function Approximation Theorem (for Policy Gradient) is satisfied: **(2)**

$$\nabla_w Q(s,a;w) = \nabla_\theta \log \pi(a|s;\theta)$$

where w defines the parameters of the function approximation for the action-value function.

(c) Show that $Q(s,a;w)$ has zero mean for any state s, i.e. show that, $\mathbb{E}_\pi\big[Q(s,a;w)\big]$ defined as $\sum\limits_{a\in A} \pi(a|s;\theta).Q(s,a;w) = 0$. **(5)**

Solution:

(a)

$$\begin{aligned}
\log \pi(a|s;\theta) &= \theta^T.\phi(s,a) - \log\Big(\sum_{b\in A} e^{\theta^T.\phi(s,b)}\Big) \\
\therefore\ \frac{\delta \log \pi(a|s;\theta)}{\delta\theta_i} &= \phi_i(s,a) - \frac{\sum\limits_{b\in A} \phi_i(s,b).e^{\theta^T.\phi(s,b)}}{\sum\limits_{b\in A} e^{\theta^T.\phi(s,b)}} \\
&= \phi_i(s,a) - \sum_{b\in A} \frac{e^{\theta^T.\phi(s,b)}}{\sum\limits_{b\in A} e^{\theta^T.\phi(s,b)}}.\phi_i(s,b) \\
&= \phi_i(s,a) - \sum_{b\in A} \pi(b|s;\theta).\phi_i(s,b) \\
&= \phi_i(s,a) - \mathbb{E}\big[\phi_i(s,\cdot)\big]
\end{aligned}$$

Therefore, $\nabla_\theta \log \pi(a|s;\theta) \quad = \quad \phi(s,a) - \mathbb{E}\big[\phi(s,\cdot)\big]$

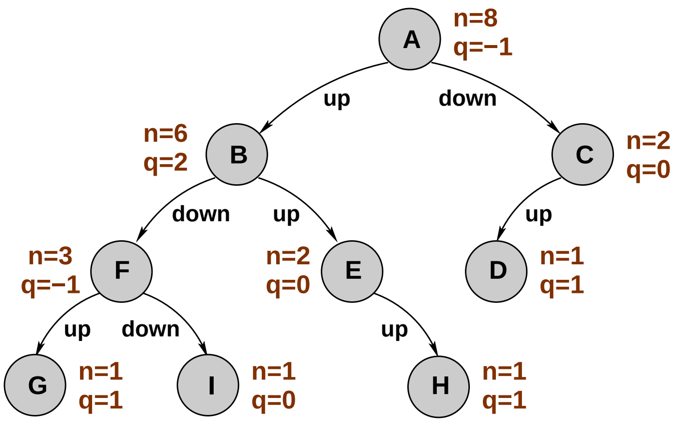
(b) To satisfy the key constraint $\nabla_w Q(s,a;w) = \nabla_\theta \log \pi(a|s;\theta)$ of the Compatible Function Approximation Theorem, we let the features of $Q(s,a;w)$ be $\nabla_\theta \log \pi(a|s,\theta)$ and we set $Q(s,a;w)$ to be linear in these features:

$$Q(s,a;w) = w^T.\nabla_\theta \log \pi(a|s;\theta)$$

(c) Finally,

$$\begin{aligned}
\sum_{a\in A} \pi(a|s;\theta).Q(s,a;w) &= \sum_{a\in A} \pi(a|s;\theta).w^T.\nabla_\theta \log \pi(a|s;\theta) \\
&= \sum_{a\in A} w^T.\nabla_\theta \pi(a|s;\theta) \\
&= w^T.\nabla_\theta\Big(\sum_{a\in A} \pi(a|s;\theta)\Big) \\
&= w^T.\nabla_\theta 1 \quad = 0 \qquad\qquad \text{[Proved]}
\end{aligned}$$

Q6. **[13 Marks]** You are using Monte Carlo Tree Search (MCTS) to decide on the next action for a two-person competitive game with 2 actions at each state (**up** and **down**). It is Player-1's turn to play in state A. The state expansion of the tree so far is as follows (each node consists of state identifier, n-value, and q-value):



Recall that, the formula for the UCT value for a node, v, is:

$$UCT(v) = \frac{q(v)}{n(v)} + c\sqrt{\frac{\ln n(parent(v))}{n(v)}}$$

where, $n(v)$ and $q(v)$ denote the n-value and q-value for node v, and $parent(v)$ indicates the parent node of v.
Assume that, the constant c in the UCT formula is given as, $c = \frac{1}{2}$.

Answer the following:

(a) What is the node that is next selected? Show your calculations/work in details. **(5)**

(b) What are the details of the node that gets expanded from the selected node? **(1)**

(c) Assuming that the simulation (rollout) from the expanded node gives a value of 1 (that is, Player-1 wins), show how the backup for that value is calculated to all of the affected nodes. **(4)**

(d) Assume that after this final rollout, we have run out of time to run the MCTS simulation and must now choose an action for Player-1 from state A. What action will be chosen and why? **(3)**

Solution:

(a) We start at the root node, A.

Since A is fully expanded, we use the UCT formula for its children, B and C.

$$UCT(B) = \frac{2}{6} + \frac{1}{2}\sqrt{\frac{\ln 8}{6}} = 0.63 \qquad \text{and} \qquad UCT(C) = \frac{0}{2} + \frac{1}{2}\sqrt{\frac{\ln 8}{2}} = 0.51$$

Since $UCT(B) > UCT(C)$, we move to node B.

Since B is fully expanded, we use the UCT formula for its children, F and E.

$$UCT(E) = \frac{0}{2} + \frac{1}{2}\sqrt{\frac{\ln 6}{2}} = 0.47 \qquad \text{and} \qquad UCT(F) = \frac{-1}{3} + \frac{1}{2}\sqrt{\frac{\ln 6}{3}} = 0.05$$

Since $UCT(E) > UCT(F)$, we move to node E.

Since E is not fully expanded, *E is the next selected node.*

(b) We shall add a new node with action **down** to E (call it J) with $n = 0$ and $q = 0$.

(c) Since A is a state reached from Player-2 playing, B is a state reached from Player-1 playing, then E is a state reached from Player-2 playing, and J is a state reached from Player-1 playing.

Thus, we shall increment q for states B and J, and decrement it for states A and E. We shall increment n for states A, B, E, and J. So, the new values are as folows:

$$\begin{array}{lcl} A & : & n = 9, \;\; q = -1 - 1 = -2 \qquad B \;:\; n = 7, \;\; q = 2 + 1 = 3 \\ E & : & n = 3, \;\; q = \;\; 0 - 1 = -1 \qquad J \;:\; n = 1, \;\; q = 0 + 1 = 1 \end{array}$$

(d) If we are using highest n-value, we'll choose action **up**, since that leads to state B and $n(B) > n(C)$.

If we are using highest $\frac{q}{n}$-value, we'll also choose action **up**, since that leads to state B and $\frac{q(B)}{n(B)} > \frac{q(C)}{n(C)}$.