CS 747 (Spring 2025) End-semester Examination

9.00 p.m. – 12.00 p.m., April 24, 2025, LA 001 and LT 004

Name: ______________________________ Roll number: ______________

Note. There are five questions in this test. Provide your answer to each question in the space following the question (and *before* the next question if one exists). You can use any blank space in the paper for rough work by drawing a line (either vertical or horizontal), writing "Rough work" on one side of it, and using the demarcated space for rough work. In addition, the remainder of this page as well as five separate pages (4, 7, 10, 13, 16) have been provided for rough work.

Question	**Marks**	
1		/4
2		/4
3		/4
4		/5
5		/8
Total:		/25

Rough work ↓

Question 1. In an n-armed bandit, $n \geq 2$, each arm yields Bernoulli rewards. Let the arms' means be $p_1, p_2, \ldots, p_n$. A designer aims to implement Thompson Sampling on this bandit instance. Interestingly, based on domain knowledge, the designer knows that each arm's mean comes from a finite set $Q = \{q_1, q_2, \ldots, q_m\}$, $m \geq 2$, where $q_i \in (0, 1)$ for $1 \leq i \leq m$. The designer knows Q. Of course, the designer does not know which element of Q is the mean for any given arm.

As an illustration of the setting described, consider that there are $n = 3$ arms, and that the designer knows that each arm's mean must be exactly 0.1, 0.4, 0.6, 0.83, or 0.87. In this case $m = 5$ and $Q = \{0.1, 0.4, 0.6, 0.83, 0.87\}$.

Write down an implementation of Thompson Sampling in which the designer's belief distribution for each arm is over Q, rather than over $[0, 1]$, as is most commonly taken. Assume a suitable initial belief, and write down pseudocode for sampling and updating beliefs that would be consistent with the principle underlying Thompson Sampling. Use any suitable notation, but describe the relevant quantities in words. For example, use lines such as "let r denote the reward obtained."

Note that this question is specifically about Thompson Sampling; you will *not* receive marks for adapting other algorithms (such as UCB or ϵ-greedy) to incorporate the knowledge of Q. [4 marks]

Answer. Since we know that each mean comes from the set Q, we maintain a belief over Q for each arm. Let the set of arms be A. We initialise belief with a uniform distribution w^0: For $a \in A$, $q \in Q$,

$$w^0_{q,a} = \frac{1}{|Q|} = \frac{1}{m}.$$

The main question is how the belief of arm a must be updated when it receives a reward. We apply the principle that "posterior is proportional to prior times likelihood". The prior belief is already available; the likelihood that a Bernoulli variable wih mean q generates a 1-reward is q, and that it generates a 0-reward is $1 - q$. Suppose arm $a^t \in A$ is sampled at time step $t \geq 1$, and it receives reward r^t. We set for $q \in Q$,

$$w^t_{q,a^t} = \frac{w^{t-1}_{q,a^t} \cdot q^{r^t} \cdot (1-q)^{1-r^t}}{\sum_{\bar{q} \in Q} w^{t-1}_{\bar{q},a^t} \cdot \bar{q}^{r^t} \cdot (1-\bar{q})^{1-r^t}}.$$

Since no data has been made available for arms other than a^t, we carry over the beliefs for them from $t - 1$ to t. For $a \in A \setminus \{a^t\}$ and $q \in Q$:

$$w^t_{q,a} = w^{t-1}_{q,a}.$$

What remains to be specified is the sampling strategy, which is the same as in Thompson Sampling: we sample x^t_a from the belief distribution of each arm $a \in A$, and pull an arm for which x^t_a is maximum. Note that x^t_a will be an element of Q. In case of ties, we perform arbitrary tiebreaking.

The pseudocode below summarises our implementation.

```
//Initialise beliefs.
For $a \in A, q \in Q$:
    $w^0_{q,a} \leftarrow \frac{1}{m}$.
For $t = 1, 2, 3, \ldots$:
    //Select an arm to pull.
    For $a \in A$:
        $x^t_a \sim w^{t-1}_{\cdot,a}$.
    $a^t \leftarrow \text{argmax}_{a \in A}\, x^t_a$.
    //Update beliefs.
    $w^t_{q,a^t} \leftarrow \frac{w^{t-1}_{q,a^t} \cdot q^{r^t} \cdot (1-q)^{1-r^t}}{\sum_{\bar{q} \in Q} w^{t-1}_{\bar{q},a^t} \cdot \bar{q}^{r^t} \cdot (1-\bar{q})^{1-r^t}}$.
    For $a \in A \setminus \{a^t\}$:
        $w^t_{q,a} \leftarrow w^{t-1}_{q,a}$.
```

Rough work ↓

Question 2. Let $M = (S, A, T, R, \gamma)$ be a continuing MDP, with notation as usual, and with $\gamma \in (0, 1)$. Your task is to construct an *episodic* MDP $M' = (S', A', T', R', \gamma')$ with the following properties.

1. $S' = S \cup \{s_\top\}$ (M' has the same states as M, and additionally a terminal state $s_\top$).
2. $A' = A$ (M' has the same actions as M).
3. $\gamma' = 1$ (there is no discounting in M').
4. For every $s \in S$ and $\pi : S \to A$,
$$V^\pi_M(s) = V^\pi_{M'}(s),$$
where the subscript denotes the MDP on which the value is defined.

Since S', A', and γ' are already given to you, it only remains for you to define T' and R'. The fourth condition essentially conveys that any discounted continuing MDP M can be implemented as an undiscounted episodic MDP M', in the sense that every policy will have the same value on both MDPs. Note that this means that an optimal policy for M' will also be an optimal policy for M, and vice versa.

You must define T' and R' in terms of the components of M, and also give a proof that the fourth condition is achieved by your construction. [4 marks]

Answer. We start thinking backwards from the goal we want to achieve: of getting $M^{\{'\}}$ to have the same value function as M. An easy way of ensuring this is by getting both M and M' to have the same Bellman equations for each policy. Note that on M, for policy $\pi : S \to A$ and state $s \in S$,

$$V^\pi_M(s) = \sum_{s' \in S} T(s, \pi(s), s')\{R(s, \pi(s), s') + \gamma V^\pi_M(s')\}. \tag{1}$$

- Since M' is undiscounted, how do we get "γTV" on the right hand side? The natural approach is to set T' to be the product of γ and T. That is, we set
$$T'(s, a, s') = \gamma T(s, a, s')$$
for $s, s' \in S, a \in A$.
- The operation above does ensure that the elements of T' we have set are non-negative, but they do not sum up to 1. In fact, $\sum_{s' \in S} T'(s, a, s') = \gamma \sum_{s' \in S} T'(s, a, s') = \gamma$. To make T' a valid distribution, we use the terminal state. For $s \in S, a \in A$, we set
$$T'(s, a, s_\top) = 1 - \gamma.$$
- Since T' is γT, a consequence is that the $T'R$ product in (1) has now become γTR. However, since we have control of R', we can set
$$R'(s, a, s') = \frac{R(s, a, s')}{\gamma}$$
for $s, s' \in S, a \in A$. We also have to set a reward for transitioning to $s_\top$, which it seems appropriate to leave as 0: that is, for $s \in S, a \in A$,
$$R'(s, a, s_\top) = 0.$$

Taken together, do these definitions of T' and R' satisfy our requirements? First, as we have verified above, $T'(s, a, \cdot)$ is indeed a probability distribution for each $s \in S, a \in A$, and also, $R'(s, a, s')$ is well-defined for every $s, s' \in S, a \in A$. To verify if policies have the same values under M' as they do under M, we write out the Bellman equations for M'. For $\pi : S \to A$ and $s \in S$,

$$\begin{aligned}
V^{\pi}_{M'}(s) &= \sum_{s' \in S \cup \{s_\top\}} T'(s, \pi(s), s')\{R'(s, \pi(s), s') + \gamma' V^{\pi}_{M'}(s')\}. \\
&= \sum_{s' \in S} T'(s, \pi(s), s')\{R'(s, \pi(s), s') + \gamma' V^{\pi}_{M'}(s')\} + T(s, \pi(s), s_\top)\{R(s, \pi(s), s_\top) + \gamma' V^{\pi}_{M'}(s_\top)\}. \\
&= \sum_{s' \in S} T'(s, \pi(s), s')\{R'(s, \pi(s), s') + \gamma' V^{\pi}_{M'}(s')\} + (1 - \gamma)\{0 + 1 \cdot 0\}. \\
&= \sum_{s' \in S} T'(s, \pi(s), s')\{R'(s, \pi(s), s') + \gamma' V^{\pi}_{M'}(s')\}. \\
&= \sum_{s' \in S} T'(s, \pi(s), s')\{R'(s, \pi(s), s') + V^{\pi}_{M'}(s')\}. \\
&= \sum_{s' \in S} \gamma T(s, \pi(s), s') \left\{ \frac{R(s, \pi(s), s')}{\gamma} + V^{\pi}_{M'}(s') \right\}. \\
&= \sum_{s' \in S} T(s, \pi(s), s')\{R(s, \pi(s), s') + \gamma V^{\pi}_{M'}(s')\}. \qquad (2)
\end{aligned}$$

We observe that M' and M have an identical set of Bellman equations ((1) and (2)) for each policy $\pi : S \to A$. Since the equations have a unique solution, it follows that $V^{\pi}_{M} = V^{\pi}_{M'}$.

Rough work ↓

Question 3. Consider a continuing MDP $M = (S, A, T, R, \gamma)$, with notation as usual, and with $\gamma \in (0, 1)$. Let $\pi_0 : S \to A$ be an arbitrary policy for M; denote its value function V^{π_0} by V_0. In this question, we compare the progress made by two different methods for iterating from V_0 (or π_0) to the optimal value function (or an optimal policy).

1. The first method is value iteration (VI), which we have covered in class. We use the notation $V_0 \xrightarrow{VI} V_1$ to mean that the vector V_1 is obtained by performing a single step of value iteration to V_0. As you are aware, VI is identical to the Bellman optimality operator.
2. The second method is a greedy variant of policy iteration (GPI), in which the policy π_1 obtained from π_0 satisfies
$$\pi_1(s) = \operatorname*{argmax}_{a \in A} Q^{\pi_0}(s, a)$$
for $s \in S, a \in A$, with ties broken arbitrarily. We denote the above operation as $\pi_0 \xrightarrow{GPI} \pi_1$. Again, as you know, GPI is a particular (greedy) form of policy improvement.

We similarly obtain V_2 by applying VI to V_1, and we obtain π_2 by applying GPI to π_1. In summary,

$$V_0 \xrightarrow{VI} V_1 \xrightarrow{VI} V_2;$$
$$\pi_0 \xrightarrow{GPI} \pi_1 \xrightarrow{GPI} \pi_2.$$

Recall that $V^{\pi_0} = V_0$. Answer the following questions.

3a. Can we conclude that $V^{\pi_1} \succeq V_1$? If you claim yes, provide a proof. If you claim no, provide a counterexample. [2 marks]

3b. Can we conclude that $V^{\pi_2} \succeq V_2$? If you claim yes, provide a proof. If you claim no, provide a counterexample. [2 marks]

For both parts, you are free to use any results we have established in class.

Answer. We claim yes for both parts a and b. For part a, we use the fact that $\pi_1 \succ \pi_0$, which must be true since $\pi_0 \xrightarrow{GPI} \pi_1$. We also use Bellman equations and a definition of the Bellman optimality operator. For $s \in S$,

$$\begin{aligned}
V^{\pi_1}(s) &= \sum_{s' \in S} T(s, \pi_1(s), s')\{R(s, \pi_1(s), s') + \gamma V^{\pi_1}(s')\} \\
&\geq \sum_{s' \in S} T(s, \pi_1(s), s')\{R(s, \pi_1(s), s') + \gamma V^{\pi_0}(s')\} \\
&= Q^{\pi_0}(s, \pi_1(s)) \\
&= \max_{a \in A} Q^{\pi_0}(s, a) \\
&= \max_{a \in A} \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V_0(s')\} \\
&= V_1(s).
\end{aligned}$$

For part b, we proceed similarly, using the fact that $\pi_2 \succ \pi_1$. For $s \in S$,

$$\begin{aligned}
V^{\pi_2}(s) &= \sum_{s' \in S} T(s, \pi_2(s), s')\{R(s, \pi_2(s), s') + \gamma V^{\pi_2}(s')\} \\
&\geq \sum_{s' \in S} T(s, \pi_2(s), s')\{R(s, \pi_2(s), s') + \gamma V^{\pi_1}(s')\} \\
&= Q^{\pi_1}(s, \pi_2(s)) \\
&= \max_{a \in A} Q^{\pi_1}(s, a) \\
&= \max_{a \in A} \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V^{\pi_1}(s')\}.
\end{aligned}$$

Now, using the result from part a, we have for $s \in S$:

$$\begin{aligned}
V^{\pi_2}(s) &\geq \max_{a \in A} \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V^{\pi_1}(s')\} \\
&\geq \max_{a \in A} \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V_1(s')\} \\
&= V_2(s).
\end{aligned}$$

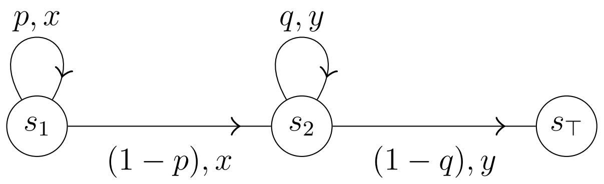
These same inequalities can also be shown on vectors by applying the Bellman operator for different policies and using the fact that it preserves the $\succeq$ relation.

Rough work ↓

Question 4. The figure below is the state-transition diagram for an episodic MDP that has non-terminal states s_1 and s_2, and a terminal state $s_\top$. When policy π is executed:

- s_1 transitions to itself with probability $p \in (0, 1)$ and to s_2 with probability $1 - p$;
- s_2 transitions to itself with probability $q \in (0, 1)$ and to $s_\top$ with probability $1 - q$.

Any transition beginning at s_1 gets a reward of $x \in \mathbb{R}$, while any transition beginning at s_2 gets a reward of $y \in \mathbb{R}$. There is no discounting. In the figure below, arrows are annotated with "transition probability under π, reward".



Suppose a single episode is executed on the MDP, with the agent starting at s_1.

4a. Let F denote the estimated value of s_1 under π from this episode, using the *first-visit* Monte Carlo estimation technique. What is $\mathbb{E}[F]$—that is, the expected value of F? [1 mark]

4b. Let E denote the estimated value of s_1 under π from this episode, using the *every-visit* Monte Carlo estimation technique. What is $\mathbb{E}[E]$—that is, the expected value of E? [4 marks]

Answer. Any episode started at s_1 will see m occurrences of s_1, followed by n occurrences of s_2, before termination, where $m \geq 1$ and $n \geq 1$. The probability of such an "(m, n) episode" is precisely

$$\mathbb{P}_{m,n} = p^{m-1}(1-p)q^{n-1}(1-q).$$

On an (m, n) episode, we have

$$\begin{aligned}
F &= mx + ny, \text{ and} \\
E &= \frac{1}{m}\{(mx + ny) + ((m-1)x + ny) + ((m-2)x + ny) + \cdots + (x + ny)\} \\
&= \frac{m+1}{2}x + ny.
\end{aligned}$$

Notice that to get E, we average the return after every visit to s_1—there are m visits. All that remains is to compute expectations of F and E by considering all possible values of m and n. For part a, we get

$$\begin{aligned}
\mathbb{E}[F] &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \mathbb{P}_{m,n}(mx + ny) \\
&= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} p^{m-1}(1-p)q^{n-1}(1-q)(mx + ny) \\
&= (1-p)(1-q)\left\{\sum_{m=1}^{\infty} p^{m-1}(mx)\sum_{n=1}^{\infty} q^{n-1} + \sum_{n=1}^{\infty} q^{n-1}(ny)\sum_{m=1}^{\infty} p^{m-1}\right\} \\
&= \frac{x}{1-p} + \frac{y}{1-q}
\end{aligned}$$

For part b, we get

$$\begin{aligned}\mathbb{E}[E] &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\mathbb{P}_{m,n}\left(\frac{m+1}{2}x+ny\right)\\ &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\mathbb{P}_{m,n}\left(m\frac{x}{2}+ny\right)+\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\mathbb{P}_{m,n}\frac{x}{2}\end{aligned}$$

The first term is the same expression as $\mathbb{E}[F]$, except with $\frac{x}{2}$ in place of x. The second term is simply $\frac{x}{2}$ since $\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\mathbb{P}_{m,n}$ is 1. Thus, we have

$$\mathbb{E}[E] = \frac{x}{2}\left(\frac{1}{1-p}\right)+\frac{y}{1-q}+\frac{x}{2} = \frac{x(2-p)}{2(1-p)}+\frac{y}{1-q}.$$

Rough work ↓

Question 5. This question has six parts: (a) through (f). Answer all parts.

5a. Does Sarsa perform *on-policy* or *off-policy* updates? Does Expected Sarsa perform *on-policy* or *off-policy* updates? [1 mark]

Answer. Both Sarsa and Expected Sarsa perform on-policy updates.

5b. State an advantage of on-line reinforcement learning over batch reinforcement learning, and also an advantage of batch reinforcement learning over on-line reinforcement learning. [1 mark]

Answer. On-line reinforcement learning requires only a constant amount of compute time per update, and it does not need to store observed transitions. On the other hand, by using memory to save experience and extracting more out of it through computationally-intensive learning updates, batch reinforcement learning usually requires many fewer samples to reach the same levels of performance on the task as on-line reinforcement learning. Batch updates also tend to be more stable for function approximation.

5c. In class we have typically taken that an agent follows a policy that specifies which action to take from each state. This encoding of behaviour is also called *closed-loop* control. By contrast, under *open-loop* control, an agent can execute a longer sequence of actions *without* sensing intermediate states. For example, if at state s at time step t, the agent can decide to execute the sequence (a_1, a_2, a_3): which means a_1 gets executed at s at time step t, then a_2 gets executed at time step $t+1$ from whichever state is reached, then a_3 gets executed at time step $t+2$ from whichever state gets reached. Thus open-loop control allows for executing a sequence of actions from each sensed state; closed-loop control is a special case in which the sequences are all of length 1.

What are the potential benefits of open-loop control? Give a real-life example that has some form of open-loop control. What kind of environments are suitable for open-loop control? [2 marks]

Answer. Sensing state can be expensive in terms of time, energy, and compute. The main advantage of open-loop control is that the agent can go without sensing state on many time steps. One practical example of an open-loop control system is an old-fashioned washing machine, which, when started, will execute a sequence such as soak for 5 minutes, then spin for 12 minutes, then mix detergent, then spin for 6 minutes, and so on, without necessarily monitoring the state of the clothes inside. Today's washing machines integrate more sensing (of temperature, weight of clothes, and so on) to control these cycles more effectively. Another example of open-loop control would be me descending a flight of stairs. At the top of the flight, I "mindlessly" step down, down, down say 5–6 times before pausing to see how many more steps are left to reach the floor. Here the high-level decision of whether to take a "down" step or a "reach the floor" step is done open loop; each physical step still takes into account balance signals from the inner ear and pressure sensations on the feet to contract and relax the body's muscles.

Open-loop control is as capable as closed loop control on MDPs with deterministic transitions. Other environments favouring open-loop controls are ones in which state changes gradually: that is, a good action for the current state is likely to also be good for the states reached within the next few time steps.

5d. A designer has implemented REINFORCE to get an agent to optimise its behaviour on a certain episodic task. The task is such that from the starting state, every policy will either reach a goal state (counting as a "successful" episode) or a bad terminal state (counting as a "failed" episode). The aim is to maximise the probability of succeeding.

The designer sets up REINFORCE to make an update to the policy parameters after each episode. However, while running it, the designer notices that after the first 10 episodes are completed, *no* policy changes have occurred; that is, the policy parameters are still those from initialisation. The update rule is correct; the learning rate is positive. What is the likely explanation for this observation? [1 mark]

Answer. The REINFORCE update rule is

$$\theta_{t+1} \leftarrow \theta_t + \alpha \sum_{t=0}^{T} G_{t:T} \nabla \ln \pi(s^t, a^t),$$

where the notation is as used in class. In the task described above, it is very plausible that the reward function is such that a non-zero reward is given only if the agent completes a successful episode. If, in the initial few episodes, the agent never gets a non-zero reward, the $G_{t:T}$ factors in the update will all be 0, and hence the weights will not change. Note that even at a local optimum (as long as the success probability is not 100%), there will be weight changes since we perform *stochastic* gradient ascent.

5e. Suppose an agent observes an "expert" who is taking actions in a known environment. The data collected by the agent is a set of state-action pairs visited by the expert. Of course, this data may only cover some part of the state space; the agent's aim is to generalise from this data to a policy that it can apply from any state.

Inverse reinforcement learning (IRL) is a popular approach for this purpose. In IRL, the agent asks: "for what reward function is the expert's behaviour optimal?" In mathematical terms, say the agent already knows (1) the environment: that is, S, A, T, γ (notation as usual), and (2) some state-action pairs that are consistent with an "optimal policy" $\pi^\star$. The agent first tries to determine a reward function R such that some $\pi^\star$ consistent with the collected data is an optimal policy for $M = (S, A, T, R, \gamma)$. Then the agent computes an optimal policy for M. R is usually represented using function approximation, taking features of states and actions as input.

What do you foresee as the primary mathematical challenge in IRL? Can you think of any alternative approaches (different from IRL) to generalise from expert-demonstration data to a full-fledged policy? [2 marks]

Answer. There are an infinite number of reward functions R for which any policy is optimal. In fact, an extreme example is a reward function that gives the same constant reward for all transitions—this reward function is optimal for every policy! The main mathematical challenge in IRL is to suitably define which among this infinite set of reward functions is more appropriate to consider as the one the expert is optimising.

A more direct way to generalise from the expert's demonstration is to use supervised learning: in fact the first stage in AlphaGo was to run supervised learning on expert games to obtain a good starting policy for subsequent optimisation (using reinforcement learning). Model-learning (as done by Ng *et al.*) is not relevant here since the model (T) is already given.

5f. Name the pioneers of reinforcement learning who were selected for the 2024 ACM A. M. Turing Award. [1 mark]

Answer. Andrew G. Barto and Richard S. Sutton.

Rough work ↓