CS 747 (Autumn 2025): End-semester Examination

1.30 p.m. – 4.30 p.m., November 22, 2025, LA 001 and LA 002

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.

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

Question 1. An undiscounted episodic MDP M is such that every policy for M is either "good" or "bad". Policies are classified good or bad depending on their value for starting state s_0: good policies have "high" values and bad policies have "low" values. Formally, for every good policy π_{g} and every bad policy π_{b},

$$V^{\pi_{\mathrm{g}}}(s_0) - V^{\pi_{\mathrm{b}}}(s_0) > \Delta,$$

where Δ is a known positive quantity. It is also known that at least a ρ-fraction of the set of policies for M are good policies, where $\rho \in (0, 1)$ is another known parameter. The set of policies is finite.

We have an agent that aims to identify a good policy for M. Unlike our usual setup, there is no requirement that the policy found be optimal; it only needs to be good. The agent does not have knowledge of the transition dynamics and rewards of M. However, the agent can interact with M by starting at s_0 and following any policy π. Upon doing so, a state-action-reward sequence is generated according to π and the transition and reward functions of M. Each reward lies in $[0, 1]$, and each episode has at most H transitions.

The agent implements the following algorithm to identify a good policy for M.

1. Select N policies uniformly at random (with replacement) from the set of all policies for M. Let these policies be $\pi_1, \pi_2, \ldots, \pi_N$.

2. For $i = 1, 2, \ldots, N$: generate L episodes by following π_i, and let V_i be the average episodic reward from these L episodes.

3. Return a policy that maximises the average episodic reward: that is, return π_i where $i = \mathrm{argmax}_{j=1}^{N} V_j$.

As you can see, the algorithm need not always return a good policy. However, the agent would like to provide a *probabilistic guarantee* that the policy returned is good. Formally, suppose the agent is given a "mistake probability" $\delta \in (0, 1)$ as input. We need to guarantee that the policy returned by the agent will be good with probability at least $1 - \delta$.

Work out the values of N and L in terms of problem parameters—Δ, ρ, H, and δ—such that the policy returned by the specified algorithm is good with probability at least $1 - \delta$. Clearly explain how your choices of N and L deliver this probabilistic guarantee. [5 marks]

Answer 1. The overall idea is that if N is large enough, then at least one good policy will be selected in Step 1 with sufficiently high probability. Then, in Step 2, if L is large enough, then the empirical values of the policies will lie sufficiently close to their true values, again with sufficiently high probability—hence, picking the empirical winner will mean selecting a good policy.

Formally, we would like N to be such that the probability of no good policy being selected in Step 1 is at most $\frac{\delta}{2}$. Thus, N must satisfy $(1-\rho)^N \leq \frac{\delta}{2}$. This is satisfied by setting

$$N = \left\lceil \frac{\ln \frac{2}{\delta}}{\log \frac{1}{1-\rho}} \right\rceil \text{ or (sufficiently and more conventionally) } N = \left\lceil \frac{1}{\rho} \ln \frac{2}{\delta} \right\rceil.$$

Now, suppose Π is a fixed set of N policies that contains a good policy. Suppose Π is the set selected in Step 1. For any bad policy in Π, by Hoeffding's inequality, the probability that its empirical value exceeds the true value by $\frac{\Delta}{2}$ or more is at most $\exp(-2L\frac{(\Delta/2)^2}{H^2})$ (observe that values are supported in $[0, H]$). Similarly, for any good policy in Π, the probability that its empirical value falls below the true value by $\frac{\Delta}{2}$ or more is at most $\exp(-2L\frac{(\Delta/2)^2}{H^2})$. Unless some bad policy exceeds or some good policy falls below, the empirical best policy must be a good policy. Hence, the probability of a good policy not being selected from Π is at most $N\exp(\frac{-L\Delta^2}{2H^2})$. By setting

$$L = \left\lceil \frac{2H^2}{\Delta^2} \ln \frac{2N}{\delta} \right\rceil,$$

we are assured that the probability of a mistake being made given Π is selected in Step 1 is at most $\frac{\delta}{2}$.

Let Π_{good} be the set of all sets of N policies that contain a good policy, and let Π_{bad} be the set of all sets of N policies that are all bad. Our overall mistake probability is

$$\begin{aligned}
\mathbb{P}\{\text{mistake}\} &= \sum_{\Pi \in \Pi_{\text{good}}} \mathbb{P}\{\text{mistake}|\Pi\}\mathbb{P}\{\Pi \text{ is selected in Step 1}\} + \sum_{\Pi \in \Pi_{\text{bad}}} \mathbb{P}\{\text{mistake}|\Pi\}\mathbb{P}\{\Pi \text{ is selected in Step 1}\} \\
&\leq \sum_{\Pi \in \Pi_{\text{good}}} \frac{\delta}{2}\mathbb{P}\{\Pi \text{ is selected in Step 1}\} + \sum_{\Pi \in \Pi_{\text{bad}}} \mathbb{P}\{\Pi \text{ is selected in Step 1}\} \\
&\leq \frac{\delta}{2} + \frac{\delta}{2} \\
&= \delta.
\end{aligned}$$

The last step is shown for full clarity on how the mistake probability is accounted for. An informal argument that corrrectly sets N to select at least one good policy and correctly sets L to identify a winner in the chosen set will be considered satisfactory.

Question 2. An agent is to take part in a lottery, which works as follows. There are m tickets available, $m \geq 1$. Each ticket $i \in \{1, 2, \ldots, m\}$ has an associated cost $c_i > 0$, and an associated probability $p_i \in [0, 1]$ of winning the lottery. The agent begins with initial funds $C > 0$, and proceeds in rounds. On each round, the agent can purchase any ticket that has not been bought earlier, and whose cost does not exceed the agent's available funds. Suppose that the agent buys ticket j. With probability p_j, the agent wins the lottery and the run ends, while with probability $1 - p_j$, the agent proceeds to the next round with its available funds decremented by c_j. If, after some rounds, the agent does not have the funds to buy any remaining ticket, then the run ends unsuccessfully for the agent. The aim of the agent is to maximise the probability of winning the lottery, while being constrained not to spend more than C.

For illustration, consider the problem instance shown on the right. One possible successful run on this instance could be (buy ticket 2 and fail, buy ticket 1 and succeed), while one possible unsuccessful run could be (buy ticket 2 and fail, buy ticket 1 and fail, buy ticket 4 and fail). Note that the agent cannot possibly buy all four tickets on any run in this instance, since their total cost exceeds C.

Ticket i	Cost c_i	Win-probability p_i
1	15	0.1
2	10	0.05
3	10	0.08
4	18	0.12
$C = 50$		

Answer the following questions for a general instance (not for the specific example shown).

2a. Suppose the agent has the knowledge of all the associated parameters: that is, the cost c_i and win-probability p_i for each ticket i, and also the initial funds C. Describe how the agent can compute an optimal way to execute its task. Recall that the goal is to maximise the probability of winning the lottery. [4 marks]

2b. Suppose the agent knows all the parameters of the instance, but not the win-probabilities $p_1, p_2, \ldots, p_m$. However, the agent is allowed to play on this fixed instance for many runs, so it can learn about the instance and use this knowledge to try to win the lottery many times. On each run the budget available is C. Briefly describe the learning approach that you would recommend for the agent, such that the agent's number of wins can be improved over relatively short horizons, and not just be asymptotically optimal. Emphasise exactly what will be learned, and how it will be put to use. 3–4 lines of qualitative description will suffice; do not specify the full implementation in detail. [1 mark]

Answer 2a. We can formulate an MDP such that an optimal policy for the MDP will translate into optimal decision-making by this agent.

- Each decision-making state s of the agent would be a proper subset of the set tickets such that the total cost of the subset c_s and the cost c_i of some remaining ticket sum to C or less. There would also be separate "success" and "failure" terminal states.
- For each state s the set of available actions would be the tickets that are not a part of s, and whose cost does not exceed the difference between C and the total cost of s. If our convention is to include all possible tickets as actions, we can set the rewards for taking illegal actions to be $-\infty$.
- Taking action i from state s will lead to terminal state "success" with probability p_i, and with the remaining probability, would go into the union of the tickets in s and i if this is a valid state, else terminate in "failure".
- The reward for reaching "success" is 1; every other reward is 0.
- There is no discounting.

The MDP defined above corresponds exactly to the process defined in the question. Hence, its solution yields optimal behaviour for the agent.

Interestingly, note that although the MDP is stochastic, the only uncertainty about the next state is whether it is terminal or not. If a transition is not to a terminal state, then the next state is fully determined: it is the union of the tickets in the current state and the one specified by the action. Hence, starting from the initial (null) state, executing an optimal policy would only require us to memorise a single sequence of tickets. Moreover, any permutation of a sequence of tickets will have exactly the same probability of succeesing on a run. Thus, it is sufficient to consider all feasible subsets of tickets that can be bought with the budget of C, and find one that maximises the probability of succeeding. If the tickets in a subset are are $i_1, i_2, \ldots, i_k$, then the probability of the subset succeeding is is $1 - \prod_{j=1}^{k}(1 - p_{i_j})$. In summary, we can enumerate all qualifying subsets, calculate the win probability for each, and pick a maximising one. On any run, the agent can play the tickets from that subset in any sequence.

Answer 2b. The unknown parameters are the win probabilities $p_1, p_2, \ldots, p_m$. The ideal learning algorithm will estimate these by sampling, and exploit based on the current estimates. Like in the model-based algorithm presented in class, we could pick a qualifying ticket uniformly at random with probability ϵ, and with the remaining probability, play according to an optimal policy for the current empirical model.

It would *not* be efficient to treat each Q-value in the MDP defined above as a separate parameter and estimate it, say, using an on-line learning method such as Q-learning. There are only m free parameters, but an exponential-in-m number of state-action pairs.

Question 3. This question has four parts: (a) through (d). Answer all four parts.

3a. In the context of generalisation and function approximation, why is tile coding considered a *linear* scheme? On the other hand, in what sense can tile coding also be interpreted as *non-linear*? [1 mark]

3b. Consider the n-step bootstrapping algorithm TD(n), where $n = 1, 2, 3, \ldots$, and also consider the more commonly-used bootstrapping algorithm TD(λ), where $\lambda \in [0, 1]$. In terms of computation and memory, what is the major differentiating factor between these two algorithms? [1 mark]

3c. Consider a continuing MDP $M = (S, A, T, R, \gamma)$, with notation as usual. Let $\pi : S \to A$ be some deterministic policy for this MDP. Now, let $\mathcal{M}'$ be the set of all MDPs of the form $M' = (S, A, T, R', \gamma)$, wherein each reward assigned by R' is either 0 or 1. Observe that every MDP in $\mathcal{M}'$ shares S, A, T, and γ with M; the only possible difference is the binary reward structure in the MDPs of $\mathcal{M}'$. Does $\mathcal{M}'$ necessarily contain an MDP for which π is the unique optimal policy? Answer yes or no, and justify your answer. [1 mark]

3d. Provide the update rule for and explain the Expected Sarsa learning algorithm. Can Expected Sarsa achieve convergence to an optimal policy? [2 marks]

Answer 3a. Under tile coding, the function is expressed as a linear combination of features. For example, if the action value function is approximated, it would be of the form

$$Q(s,a) \approx w_1\phi_1(s,a) + w_2\phi_2(s,a) + \cdots + w_m\phi_m(s,a),$$

where the ϕ's are binary features and the w's are the weights being learned. Due to this architecture, the method would enjoy certain theoretical guarantees, such as the convergence of linear TD(λ).

However, the ϕ's are not necessarily the raw features provided by the designer. For example, the application could have a real-valued feature $x(s,a)$, which gives rise to a number of binary features $\phi_i(s,a)$, called tiles. For example, we could set

$$\phi_i(s,a) = 1 \text{ if and only if } \frac{x(s,a)}{\text{tile width}} \in (i, i+1].$$

Notice that while it is linear in the ϕ's, the approximation is *non-linear* in the x's.

Answer 3b. To implement TD(n), the agent needs to keep the preceding n transitions in memory. However, to implement TD(λ), it needs to maintain an eligibility trace, comprising one real value for each state. The latter is usually preferred in practice, since it facilitates a recursive implementation that can also be adapted to function approximation.

Answer 3c. Yes. Consider $M' \in \mathcal{M}'$, where $M' = (S, A, T, R', \gamma)$, in which for $(s, a, s') \in S \times A \times S$:

$$R'(s,a,s') = \begin{cases} 1 & \text{if } \pi(s) = a, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, π gives a value of $\frac{1}{1-\gamma}$ from each state, since it receives a reward of 1 at each time step. Every other policy will receive a 0-reward from at least one state, and at most 1 in every state, and hence have a strictly dominated value function.

Answer 3d. Expeccted Sarsa is an on-policy, on-line learning algorithm for the controls of MDPs. As the agent goes through its life, suppose that it follows policy π^t at time step t. Expected Sarsa can be executed to work for any choice of π^t, but it is commonly an ϵ–greedy policy with respect to the action value function being learned. Now, suppose the agent encounters a transition (s^t, a^t, r^t, s^{t+1}). At time step $t+1$, the agent performs the following learning update under Expected Sarsa.

$$Q(s^t,a^t) \leftarrow Q(s^t,a^t)(1-\alpha) + \alpha\left(r^t + \sum_{a\in A} \pi^{t+1}(s^{t+1},a)Q^t(s^{t+1},a)\right),$$

where α is a learning rate. If the exporation and learning rates are both *annealed*, say as $\frac{1}{t+1}$, indeed Expected Sarsa will converge to the optimal action values and induce an optimal policy.

Question 4. An agent is set up to execute experience replay on an MDP that has states s_1, s_2, and actions a_1, a_2. The agent has collected the batch of four transitions shown in the table below.

Transition number	State	Action	Reward	Next State
1	s_1	a_1	2	s_1
2	s_1	a_1	0	s_2
3	s_2	a_2	3	s_2
4	s_2	a_1	1	s_1

The agent replays the transitions in round robin, in the sequence 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, ... The discount factor used in $\gamma = \frac{2}{3}$. All the entries of the Q-table are initialised to 0, and the update rule used is that of Q-learning.

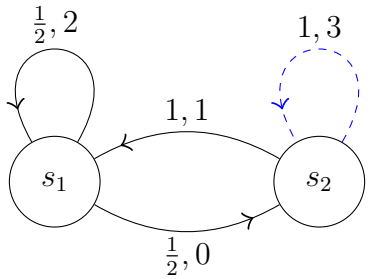
4a. Suppose the agent uses a constant learning rate of $\alpha = 0.5$. What is the Q-table after the first pass is completed over the set of transitions—that is, after the first four updates are made? [2 marks]

4b. Suppose the agent uses a learning rate that is annealed suitably (say set to $1/(i+1)$ for the i-th pass). To what values will the entries in the Q-table converge as more and more passes are made over this batch of transitions? [3 marks]

Answer 4a. Each update applies to only one of the Q-values; the others remain unchanged.
The first update sets $Q(s_1, a_1)$ to $Q(s_1, a_1)(1-\alpha) + \alpha(2 + \gamma \max\{Q(s_1, a_1), Q(s_1, a_2)\} = 1$.
The second update sets $Q(s_1, a_1)$ to $Q(s_1, a_1)(1-\alpha) + \alpha(0 + \gamma \max\{Q(s_2, a_1), Q(s_2, a_2)\} = \frac{1}{2}$.
The third update sets $Q(s_2, a_2)$ to $Q(s_2, a_2)(1-\alpha) + \alpha(3 + \gamma \max\{Q(s_2, a_1), Q(s_2, a_2)\} = \frac{3}{2}$.
The fourth update sets $Q(s_2, a_1)$ to $Q(s_2, a_1)(1-\alpha) + \alpha(1 + \gamma \max\{Q(s_1, a_1), Q(s_1, a_2)\} = \frac{2}{3}$.
The table below summarises the Q-table after each update is made.

	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_2, a_1)$	$Q(s_2, a_2)$
Initially	0	0	0	0
After first update	1	0	0	0
After second update	$\frac{1}{2}$	0	0	0
After third update	$\frac{1}{2}$	0	0	$\frac{3}{2}$
After fourth update	$\frac{1}{2}$	0	$\frac{2}{3}$	$\frac{3}{2}$

Answer 4b. Since (s_1, a_2) is never visited, its Q-value remains untouched. The other Q-values converge to the optimal action values ($Q^\star$ values) of the MDP shown below. Arrows are annotated with "probability, reward". Action a_1 is solid, black, and action a_2 is dashed, blue. Note that $\gamma = \frac{2}{3}$.



Upon calculating for this MDP, we obtain $Q^\star(s_2, a_2) = 9; Q^\star(s_1, a_1) = 6; Q^\star(s_2, a_1) = 5$. Hence, in summary, the eventual values in our Q-table are:

$$Q(s_1, a_1) = 6; Q(s_1, a_2) = 0; Q(s_2, a_1) = 5; Q(s_2, a_2) = 9.$$

Question 5. Consider the zero sum matrix game shown on the right, with two players: A (row) and B (column). A can take actions a_1, a_2, and a_3, while B can take actions b_1 and b_2. Each entry in the matrix shows A's reward when A and B play actions from the corresponding row and column, respectively (B gets the negative of the same reward).

	b_1	b_2
a_1	3	6
a_2	5	0
a_3	2	5

5a. Compute minimax-optimal strategies for A and B, and also provide the value for each player when they play these strategies against each other. [4 marks]

5b. Suppose B plays a strategy that picks each action with equal probability (that is, b_1 with probability 50% and b_2 with probability 50%). What is the maximum value (that is, expected reward) that A can achieve against this strategy of B? How must A play to achieve this value? [1 mark]

Answer 5a. Suppose that A plays the strategy (p_1, p_2, p_3), where $p_1, p_2, p_3 \in [0, 1]$ and $p_1 + p_2 + p_3 = 1$. It is apparent by looking at the reward matrix that for A, action a_1 dominates action a_3 whatever be the action of B (numerically, $3 > 2$ and $6 > 5$). Hence, a minimax-optimal strategy for A must have $p_3 = 0$. If A plays $p = (p_1, 1 - p_1, 0)$ and B plays $q = (q_1, 1 - q_1)$, the expected reward for A is

$$\begin{aligned} R_{\mathrm{A}}(p, q) &= 3p_1q_1 + 6p_1(1 - q_1) + 5(1 - p_1)q_1 + 0(1 - p_1)(1 - q_1) \\ &= -8p_1q_1 + 6p_1 + 5q_1 \\ &= q_1(5 - 8p_1) + 6p_1. \\ &= p_1(6 - 8q_1) + 5q_1 \end{aligned}$$

If $5 - 8p_1$ is positive, B will set $q_1 = 0$ to maximise its own reward, and the resulting reward for A would be $6p_1$. If $5 - 8p_1$ is non-positive, B will set $q_1 = 1$ to maximise its own reward, and the resulting reward for A would be $5 - 2p_1$. It is seen that A has the highest assured reward by playing $p_1 = \frac{5}{8}$; the corresponding reward is $\frac{15}{4}$.

If $6 - 8q_1$ is positive, A will set $p_1 = 1$ to maximise its own reward, and the resulting reward for A would be $6 - 3q_1$. If $6 - 8q_1$ is non-positive, A will set $p_1 = 0$ to maximise its own reward, and the resulting reward for A would be $5q_1$. It is seen that B has the highest assured reward by playing $q_1 = \frac{3}{4}$; the corresponding reward is $-\frac{15}{4}$.

In summary, A's minimax-optimal strategy is $(\frac{5}{8}, \frac{3}{8}, 0)$, while B's minimax-optimal strategy is $(\frac{3}{4}, \frac{1}{4})$. When both players use these strategies, A gets an expected reward of $\frac{15}{4}$ and B the negative of the same.

Answer 5b. For the same reasons as in 5a, we conclude that A will not play action a_3. Given B's strategy, we have $R_A(p, q) = 2p_1 + 2.5$, which is maximised by setting $p_1 = 1$. Thus, A must deterministically play a_1, which will give it an expected reward of 4.5.