CS 747 (Autumn 2016): End-semester Examination

Instructor: Shivaram Kalyanakrishnan

9.30 a.m. – 12.30 p.m., November 17, 2016, 101/103/105 New CSE Building

Total marks: 25

Note. Provide brief justifications and/or calculations along with each answer to illustrate how you arrived at the answer.

Question 1. What is the difference between *planning* and *learning*? When we considered the learning problem in class, what assumption did we make regarding the underlying MDP, which we did not have to make for solving the planning problem? [2 marks]

Question 2. An agent interacts with an episodic MDP M, which has a fixed start state s_{init}. In other words: each episode starts in s_{init}, and regardless of the agent's actions, the episode terminates with probability 1 after a finite number of transitions. In fact, it is guaranteed that there will be no more than H transitions in each episode, for some $H > 0$. The agent's rewards are bounded in the range $[-R_{\max}, R_{\max}]$, for some $R_{\max} > 0$. No discounting is used for calculating values (that is, the discount factor is 1).

If $V^{\star}_{M}(s_{init})$ is the value of s_{init} under an optimal policy for M, clearly the maximum expected cumulative reward that an agent can accrue in $J > 0$ episodes—by following an optimal policy—is $V^{\star}_{M}(s_{init}) \cdot J$. On the other hand, a *learning algorithm* $\mathcal{L}$, which cannot know the optimal policy for an arbitrary MDP, can only hope to achieve optimal behaviour in the limit of experience. The **expected cumulative regret** of $\mathcal{L}$ on M (after J episodes) is the difference between $V^{\star}_{M}(s_{init}) \cdot J$ and the expected sum of rewards that $\mathcal{L}$ accrues over J episodes while interacting with M.

Describe a learning algorithm whose expected cumulative regret scales *logarithmically* with J on every episodic MDP M. The learning algorithm is given the number of states and actions in the MDP as input. The transition and reward functions are not known directly, but the agent can sample them by interacting with M. The agent has knowledge of H and $R_{\max}$. Provide a bound on the expected cumulative regret, which, in addition to showing the logarithmic dependence on J, also shows the dependence on M, H, and $R_{\max}$. Sketch a proof that $\mathcal{L}$ enjoys the claimed bound. [5 marks]

Question 3. Recall that Monte Carlo Tree Search is a key module in *AlphaGo*, the program that recently defeated the human Go champion. What is the primary challenge in applying Monte Carlo Tree Search to the game of carrom (that is, the simulated version of carrom on which you worked for your project)? Can the idea of model-based forward simulation still be used effectively by a carrom-playing agent? [2 marks]

Question 4. Consider the MDP shown in Figure 1, with states s_1, s_2, and s_3, and actions a_1 and a_2. The MDP is *episodic*. Each episode has an *equal* probability (1/3) of starting in any of the three states. From s_1, action a_1 deterministically takes the agent to s_2, and action a_2 deterministically takes the agent to s_3. Transitions out of s_2 and s_3 terminate the episode and reset the agent in one of the three states. No discounting is used in the calculating values. In the figure, transitions resulting from action a_1 are shown with solid arrows; those from action a_2 are shown with dotted arrows. The reward for each transition is shown along with the corresponding arrow.

The following questions, about an agent that interacts with this MDP, relate to various aspects of reinforcement learning: (a) prediction, (b) function approximation, (c) control, (d) partial observability, and (e) batch learning. Answer each question independently: do not carry over assumptions from one to the other.

4a. Suppose an agent follows policy π^{111}, which is such that $\pi^{111}(s_1) = a_1$, $\pi^{111}(s_2) = a_1$, and $\pi^{111}(s_3) = a_1$. The agent estimates the value function of π^{111} using the TD(0) algorithm. A learning rate of $\alpha_j = \frac{1}{j}$ is used during the j^{th} episode. To what will the agent's estimate converge? [1 mark]

4b. This time the agent uses a linear generalisation scheme to approximate the value function of π^{111}. It has a single scalar parameter θ; features corresponding to the states are: $\phi(s_1) = 1$, $\phi(s_2) = -1$, and $\phi(s_3) = 2$. In other words, the agent intends to approximate $V^{\pi^{111}}(s_1) \approx \theta$, $V^{\pi^{111}}(s_2) \approx -\theta$, and $V^{\pi^{111}}(s_3) \approx 2\theta$. If the agent applies TD(1) with this linear function approximation scheme, while following π^{111} and continuing to anneal the learning rate harmonically, to what will θ converge? [2 marks]

4c. Suppose the agent is interested in optimal control, and it now maintains an estimate of an action value function, which it updates using a learning rate $\alpha_j = \frac{1}{j}$ during the j^{th} episode. The agent uses an ϵ-greedy policy, with $\epsilon = \frac{1}{10}$; this exploration rate is kept constant, and not decayed over time. (i) To what will the estimate converge if the agent makes Q-learning updates? (ii) To what will the estimate converge if the agent makes Sarsa updates? [3 marks]

4d. Suppose that unfortunately, our agent loses its sensation, leaving it unable to detect in which state it currently is. Even more severely, it loses its memory: it cannot remember its previous actions and rewards. All it knows is that at every stage, it can choose either action a_1 or action a_2. What (stateless, memoryless) strategy will yield the agent the highest expected per-episode reward? Do not worry how the agent might *learn* such a strategy; assume that it (magically) knows the best way to behave in the environment given its handicaps. [2 marks]

4e. Suppose our agent decides to use a batch learning method, specifically experience replay with Q-learning updates. The agent follows some exploration policy to gather the batch of trajectories shown below (from 6 episodes).

$$[s_2, a_2, 3] \quad [s_3, a_2, 2] \quad [s_1, a_1, 4, s_2, a_2, 3] \quad [s_3, a_2, 2] \quad [s_1, a_2, 1, s_3, a_1, 5] \quad [s_1, a_1, 4, s_2, a_1, 0]$$

The agent now makes repeated passes over these samples, performing Q-learning updates with a learning rate of $\frac{1}{t}$ during the t^{th} pass. If an infinite number of passes are performed during this batch update, to what will the agent's estimate of the optimal action value function converge? [2 marks]

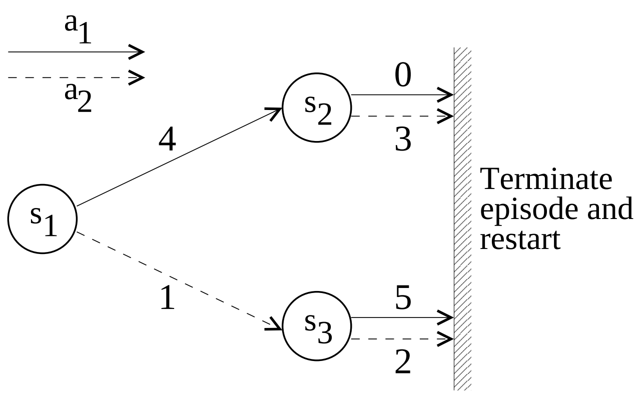

Figure 1: MDP corresponding to Question 4.

Question 5. What broad principle was illustrated in class using Tetris as a case study? What lessons from our related discussion carry over to other sequential decision making tasks? [2 marks]

Question 6. Consider the following zero sum matrix game, with two players A (row) and B (column). A can take actions a_1 and a_2, 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 (B gets the negative of the same reward).

	b_1	b_2
a_1	1	2
a_2	1	-1

Let $R_A(p, q)$ be the expected reward obtained by A when

- A plays a_1 with probability p, and action a_2 with probability $1 - p$, and
- B plays b_1 with probability q, and action b_2 with probability $1 - q$.

Since it is a zero sum game, the corresponding expected reward for B is $R_B(p, q) = -R_A(p, q)$.

6a. Plot a graph of $\min_q R_A(p, q)$ against p. [1 mark]

6b. Plot a graph of $\min_p R_B(p, q)$ against q. [1 mark]

6c. If π_A is a minimax strategy for A, and π_B is a minimax strategy for B, show that they are also best responses against each other: that is, if A (B) plays π_A (π_B), then B (A) can maximise its expected reward by playing π_B (π_A). [2 marks]

Solutions

1. The objective of both planning and learning is to find an optimal policy for a given MDP. Under planning, we assume that the MDP (including its transition and reward functions) is fully specified to the agent, and so finding an optimal policy amounts to *computing* it. In the learning setting, the agent is not directly given access to the transition probabilities and rewards, but the agent may *interact* sequentially with the MDP. In order to find an optimal policy merely by interaction, it is necessary that every state in the MDP be *reachable* from every other state.

2. Define a multi-armed bandit instance in which each arm corresponds to a fixed deterministic policy π for M. Let the "pull" of an arm correspond to playing the associated policy for one entire episode, starting at s_{init} and until the episode terminates. Clearly,

- the number of arms in this bandit is $|A|^{|S|}$,
- each arm has its episodic rewards lying in $[-HR_{\max}, HR_{\max}]$, and
- an optimal arm plays an optimal policy $\pi^\star$ for M, getting an expected reward of $V^\star_M(s_{\text{init}})$.

We can apply the UCB algorithm to our bandit instance, invoking it suitably to account for the range of rewards. From a result we know, we can say that after J pulls, the regret of our algorithm upper-bounded by

$$O\left(\sum_{\pi: V^\pi_M(s_{\text{init}}) \neq V^\star_M(s_{\text{init}})} \frac{(HR_{\max})^2}{V^\star_M(s_{\text{init}}) - V^\pi_M(s_{\text{init}})} \log(J)\right).$$

3. The two main ideas underlying Monte Carlo Tree Search are (1) to evaluate states a few steps into the future, while also accounting for the cost of reaching those states from the current one, and (2) maintaining estimates of values of visited states, and updating them after each visit. The second idea cannot be implemented as is in carrom: since it has a continuous state space and action noise, states will typically not get visited more than once. The idea of evaluating states a few steps into the future is still valid. If state-value estimation is inexact (as it invariably is in practice), one usually obtains more reliable estimates by looking a few steps into the future, rather than directly evaluating the current state.

4a. TD(0) converges to the value function of the policy being evaluated. The agent's estimate converges to

$$V^{\pi^{111}}(s_1) = 4; V^{\pi^{111}}(s_2) = 0; V^{\pi^{111}}(s_3) = 5.$$

4b. Under linear TD(1), the coefficient θ at convergence minimises the squared distance between the approximated values and the true values of the states, weighted by the corresponding stationary probabilities. When π^{111} is followed, the stationary probabilities of s_1, s_2, and s_3 are 1/4, 1/2, and 1/4, respectively. Thus, we have

$$\theta_{\text{convergeed}} = \min_{\theta \in \mathbb{R}} \left(\frac{1}{4}(4-\theta)^2 + \frac{1}{2}(0+\theta)^2 + \frac{1}{4}(5-2\theta)^2\right) = 2.$$

4c. Notice that the agent's behaviour ensures that every state-action pair gets visited infinitely often in the limit. Q-learning will converge to the optimal action value function, given by

$$Q^\star(s_1,a_1)=7; Q^\star(s_1,a_2)=6; Q^\star(s_2,a_1)=0; Q^\star(s_2,a_2)=3; Q^\star(s_3,a_1)=5; Q^\star(s_3,a_2)=2.$$

Since action selection is ϵ-greedy, the agent will eventually start taking optimal actions with probability $1-\epsilon+\frac{\epsilon}{2}$, and suboptimal actions with probability $\frac{\epsilon}{2}$. Sarsa, an on-policy method, will converge to the action value function of this converged behaviour policy B:

$$Q^B(s_1,a_1)=6.85; Q^B(s_1,a_2)=5.85; Q^B(s_2,a_1)=0; Q^B(s_2,a_2)=3; Q^B(s_3,a_1)=5; Q^B(s_3,a_2)=2.$$

4d. In class we discussed how *memory* can serve as a defence in the face of state aliasing (partial observability). The essence of this question is to highlight a second defence: *randomisation*, which can help to a lesser extent. Let us assume our agent plays action a_1 with probability p and action a_2 with probability $1-p$. The expected episodic reward of the agent is then

$$\frac{1}{3}\left(4p^2+7p(1-p)+6(1-p)(p)+3(1-p)^2\right)+\frac{1}{3}\left(3(1-p)\right)+\frac{1}{3}\left(5p+2(1-p)\right).$$

This expression is maximised neither at $p=1$ (play only a_1) nor $p=0$ (play only a_2), but at $p=\frac{7}{12}$, which corresponds to a stochastic policy.

4e. In general, batch Q-learning will converge to the action value function that is optimal with respect to the maximum-likelihood model of the data. In this case, the MDP is deterministic and indeed there is at least one of every possible transition in the data set. Consequently the batch update will yield the optimal action value function:

$$Q^\star(s_1,a_1)=7; Q^\star(s_1,a_2)=6; Q^\star(s_2,a_1)=0; Q^\star(s_2,a_2)=3; Q^\star(s_3,a_1)=5; Q^\star(s_3,a_2)=2.$$

5. We used Tetris as a case study to illustrate that when generalisation and function approximation are used, it is possible for policy search methods to outperform value function-based methods. Thus, *representation* is a key factor in deciding the success and failure of methods on different learning tasks.

6. Clearly A's expected reward is

$$R_A(p,q)=-R_B(p,q)=pq(1)+p(1-q)(2)+(1-p)q(1)+(1-p)(1-q)(-1),$$

which may be rewritten as

$$R_A(p,q)=-R_B(p,q)=3p+2q-3pq-1.$$

Based on this expression, we obtain the required plots for (a) and (b), which are shown in Figure 2. Notice that every $p\in[\frac{2}{3},1]$ yields a minimax strategy for A; the sole minimax strategy for B is obtained by playing $q=1$. Let us consider (c). If indeed A plays a minimax strategy, B's reward is $3p(q-1)-2q+1=q(3p-2)+1-3p$, which is only maximised by $q=1$. If B plays $q=1$, every strategy of A, including its minimax strategies, obtain the same expected reward of 1.

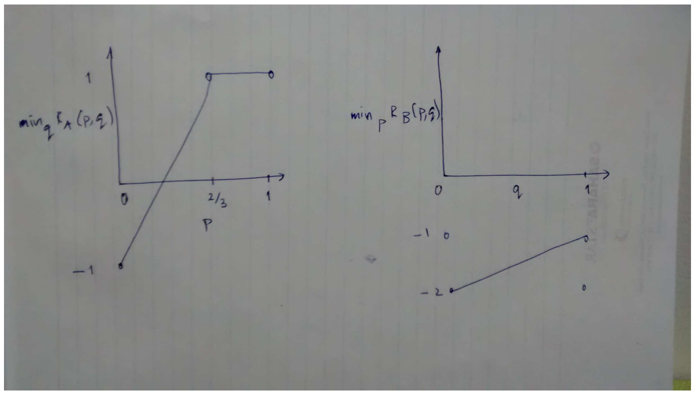

Figure 2: Plots for Question 6.