COS 435/ECE 433: Spring 2025 Midterm Solution

March 18, 2025

Questions

1. (6 points) Consider an autonomous drone trying to reach a destination while avoiding obstacles. Would you model this as a multi-armed bandit (MAB) problem or a reinforcement learning (RL) problem? Explain your answer in 1 – 2 sentences.

   It is RL. The current action affects future trajectories.

2. (12 points) Explain what distribution shift is in the context of behavioral cloning? Explain one way to mitigate its effects. (1 – 2 sentences each)

   Distribution shift is where small differences in state transitions performed by the BC policy eventually lead to larger deviations in the visited states. This could lead the BC policy to end up in states that may never have been visited by the expert, leading to poor performance.

   One way to mitigate distribution shift is

   - **Data augmentation:** Augmenting each expert state–action pair by applying controlled perturbations to the state.
   - **DART:** Injecting noise into the expert's actions during data collection so the training data covers a broader range of states.

3. (10 points) In a *finite horizon* MDP, the episode always terminates after exactly T steps. Explain why the optimal policy for a finite horizon MDP might depend on the current time step t (i.e., why you should include the integer t as an additional input to your policy). Justify your answer in 1 – 2 sentences.

   Since the process terminates after a fixed number of steps, the best action at a given state can change as fewer steps remain. The value of future rewards and the decision of which action to take varies with the time remaining before termination. This is in contrast to infinite horizon MDPs where the optimal policy can be stationary. As an example, when t is close to T, the optimal policy is to exploit more for greedy actions, as opposed to exploring more to reach the high-return states when t is small.

4. (15 points) Consider a simple MDP with states $\{s_1, s_2\}$ and actions $\{a_1, a_2\}$. Suppose the reward function is defined as $r(s, a) = 1$ if $a = a_1$ and 0 otherwise, and the transition is deterministic: $s_1 \xrightarrow{a_1} s_2$ and $s_2 \xrightarrow{a_1} s_1$, $s_1 \xrightarrow{a_2} s_1$ and $s_2 \xrightarrow{a_2} s_2$. With $\gamma = 0.9$ and a policy that always selects a_1, write the Bellman equations for $V(s_1)$ and $V(s_2)$ and solve for $V(s_1)$.

Under the given policy, the Bellman equations are:

$$V(s_1) = r(s_1, a_1) + \gamma V(s_2) = 1 + 0.9\, V(s_2),$$

$$V(s_2) = r(s_2, a_1) + \gamma V(s_1) = 1 + 0.9\, V(s_1).$$

Substituting the second equation into the first:

$$V(s_1) = 1 + 0.9\,(1 + 0.9\, V(s_1)) = 1 + 0.9 + 0.81\, V(s_1).$$

This simplifies to:

$$V(s_1) - 0.81\, V(s_1) = 1.9 \quad \Rightarrow \quad 0.19\, V(s_1) = 1.9,$$

so

$$V(s_1) = \frac{1.9}{0.19} = 10.$$

Then,

$$V(s_2) = 1 + 0.9 \times 10 = 10.$$

5. (15 points) In 1 – 2 sentences each, explain how value iteration and policy iteration work. Then name one difference between these two methods (1 sentence).

Value Iteration is a RL method that iteratively updates the value function using the Bellman optimality equation until convergence, then derives the optimal policy from the optimal value function.
Policy Iteration is a RL method that alternates between policy evaluation (computing the value function for the current policy) and policy improvement (updating the policy to be greedy with respect to the value function) until the policy stabilizes.
One important difference between these two methods is

- Value iteration implicitly maintains the policy, while policy iteration explicitly maintains and updates the policy.
- Value iteration use Bellman optimality equation while policy iteration uses Bellman equation (not with optimality).

6. (10 points) What is the purpose of using the log-likelihood substitution trick in deriving the policy gradient formula? Note that we're not asking *what* it is, but *why* it is a good idea. Your answer should be a 1 – 2 sentences.

Computing the policy gradient requires taking the gradient of the objective function with respect to the policy parameters. The gradient operator cannot be moved inside the expectation because the probability of a trajectory also depends on the policy parameters. Using this trick removes this factor from the gradient formula and adds a factor, which is the probability of a trajectory instead (not the gradient of the probability), which can be estimated via Monte Carlo sampling.

7. (6 points) Write one short equation relating $V^*(s)$ to $Q^*(s,a)$. (There are multiple correct answers.)

> $V^*(s)$ is equal to $Q^*(s,a)$ if and only if a is an optimal action at state s (i.e. a obtains the maximum Q-value of all actions available at s): $V^*(s) = \max_a Q^*(s,a)$
> We also have: $Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{p(s'|s,a)}[V^*(s')]$

8. (10 points) Given an MDP, you run Q-learning and it converges to $Q^\star(s,a)$. Is the TD error $(\frac{1}{2}(Q(s,a) - (r(s,a) + \gamma \max_{a'} Q(s',a')))^2)$ zero for all transitions (s,a,r,s')? Either provide a short proof or a counterexample.

> **No.** It will be zero in expectation, but if the environment has nondeterministic transitions then the TD error of a specific transition may be nonzero.

9. While most of this class is focused on the fully-observed settings, in this question you will work on a partially observed problem. Imagine a robot is navigating a cluttered environment where the camera can only provide a narrow view at any moment. This means that the robot cannot see the entire room, and sometimes may see the same camera input in different positions (e.g., two walls look identical if standing right in front of them).

   (10 points) Why would Q-learning fail on this problem? Explain your answer in terms of the Q-learning update rule. (2 – 4 sentences)

> The Q-learning update rule (for observation o instead of state s) is:
>
> $$Q(o,a) \leftarrow Q(o,a) + \alpha \left[r(s,a) + \gamma \max_{a'} Q(o',a') - Q(o,a) \right].$$
>
> You cannot distinguish the the two states for the same observation (in the two wall example), therefore:
>
> - 1. The reward $r(s,a)$ will be different for two states with the same observation, this causes ambiguity for $Q(o,a)$ update;
> - 2. For $Q(o',a')$ term, the next state s' after transition can be drawn from two possible distributions (one for each state s), although they share the same current observation o;
> - 3. For $\max_{a'} Q(o',a')$, one next observation o' can also have multiple underlying states s', the max operator over action is not faithful to any state s' in general.
>
> These can make Q-learning biased and unstable to converge. *(Answer 2 points from above 3, as well as the Q-learning update rule, will be granted full score)*

   (6 points) As a stubborn engineer, you insist on using Q-learning to solve this problem. How would you define the observations so that Q-learning can solve this problem? (1 – 2 sentences)

> - In the POMDP, a single camera image does not provide enough information to make optimal decisions. We can aggregate a history of observations into a single state representation with a sequence model (like a RNN) to process the images collected over time. The RNN takes in each observation and updates an internal hidden state that summarizes the recent history. This hidden state combines information from multiple time steps, capturing details about the environment that using just a single observation would miss.
> - We can also add position or location information of the robot in observation to make it not partially observable.