CS 747 (Spring 2025) Week 12 Test (Batch 1)

5.35 p.m. – 6.00 p.m., April 17, 2025, LA 001

Name: _____ Roll number: _____

Note. There are three questions, one on the first page and two on the second page. Provide your answer to each question in the space given below it. Draw a line (either vertical or horizontal) and do all your rough work on one side of it.

Question 1. The general idea behind the Dyna-Q architecture is that learning a model in addition to an action value function can help accelerate learning, since (assuming compute is available) additional Q-learning updates can be performed using transitions simulated using the model. Consider practical real-world applications, such as the soccer, helicopter control, finance, and healthcare-related ones we have discussed in class. Suppose a learning agent for these tasks indeed has sufficient time and resources for computing. Will Dyna-Q necessarily speed up the agent's learning? Why or why not? [1 mark]

Answer 1. In most practical tasks, due to the size of the state space, the learned model will have to use some form of function approximation. Consequently there is almost always some error in predicting next states. Simulation using an erroneous model could lead to systematic errors in the learned policy—hence the model could hurt, rather than benefit, performance. This is the main reason model-learning is not commonly undertaken across all applications of reinforcement learning. Where model-learning is done (such as the helicopter control work of Ng *et al.*), designers usually invest a lot of effort into making the model as accurate as possible.

Question 2. The first step in the helicopter control application of Ng *et al.* is to record trajectories of flight when the control is performed by a human pilot. Is the reward function learned from this data? If yes, provide details; if not, specify how rewards are obtained. [1 mark]

Answer 2. The reward function is *defined* by the authors. It is not induced from the flight data.

Question 3. Recall from the proof of the regret bound for UCB that the main tool used to prove that each arm's upper confidence bound is indeed a high-probability upper bound on the arm's true mean is Hoeffding's inequality. In UCT, an upper confidence bound is similarly defined for the Q-value of each state-action pair encountered within a fixed number of steps from the current state. Nonetheless, it is *not* possible to directly apply Hoeffding's inequality to argue for the correctness of the upper confidence bound on each Q-value in UCT (the argument becomes more involved). What is the main difference between the bandit and MDP settings, which is the cause for this disparity? [1 mark]

Answer 3. When UCB is applied to bandits, an upper confidence bound is constructed on the mean reward of each arm. The samples used to construct the "empirical mean" component of the upper confidence bound are i.i.d. samples drawn from the arm's reward distribution. Hoeffding's inequality is used to bound the deviation of the empirical mean from the true mean.

When UCT is applied in MDPs, the upper confidence bound is constructed on the "Q-value" of each state-action pair visited in the tree. But the Q-value under which policy?! Since the policy used for each simulation is based on the downstream upper confidence bounds, it can keep changing. For example, following (s_1, a_1), the first time a trajectory may go through (s_2, a_2), and the next time it could go through (s_2, a_3). Since the empirical returns from these trajectories are used for getting the "empirical mean" component of the upper confidence bound, it becomes the average of samples from random variables that are *not* identically distributed. Hence Hoeffding's inequality cannot be applied directly, as in bandits. However, in the long-term, the fraction of suboptimal actions taken by UCT within the tree vanishes, and so most returns come from the same distribution.

CS 747 (Spring 2025) Week 12 Test (Batch 2)

6.15 p.m. – 6.40 p.m., April 17, 2025, LA 001

Name: _____ Roll number: _____

Note. There are three questions, one on the first page and two on the second page. Provide your answer to each question in the space given below it. Draw a line (either vertical or horizontal) and do all your rough work on one side of it.

Question 1. In Experience Replay, each recorded transition is potentially used multiple times to make Q-learning updates. The information contained in any given transition (suppose it is (s, a, r, s')) remains the same; the transition itself does not change as learning updates are performed. How, then, does the update process benefit from making multiple updates using the same transition? [1 mark]

Answer 1. Let Q_0 be the initial Q-table and for $i \geq 1$, let Q_i denote the Q-table after i learning updates have been made by Experience Replay. Suppose that a particular transition is used in updates j and k with $1 \leq j < k$. Update j is

$$Q_j(s,a) \leftarrow Q_{j-1}(s,a)(1-\alpha) + \alpha(r + \gamma \max_{a'} Q_{j-1}(s',a')),$$

while update k is

$$Q_k(s,a) \leftarrow Q_{k-1}(s,a)(1-\alpha) + \alpha(r + \gamma \max_{a'} Q_{k-1}(s',a')).$$

Although the transition used is the same, notice that the update is based on bootstrapping. The j-update is based on Q_{j-1}, whereas the subsequent k-update is based on Q_{k-1}. In a rough sense, Q_i becomes more accurate as i increases. Hence, $Q_{k-1}(s,a)$ is likely to be a better approximation of $Q^\star(s,a)$ compared to $Q_{j-1}(s,a)$. In turn, $Q_k(s,a)$ is likely to be a better approximation of $Q^\star(s,a)$ than $Q_j(s,a)$.

Question 2. Is supervised learning used in any component of the solution devised by Ng *et al.* for helicopter control? If yes, provide details; if not, explain why not. [1 mark]

Answer 2. Yes: the transition model is learned using supervised learning. A data set of (state, action, next state) tripes is collected in a sequence when the helicopter is flown by a human pilot. Thereafter, locally-weighted linear regression is used to obtain a model that predicts (stochastically) next state when given state and action as input.

Question 3. In decision-time planning, what are the main tradeoffs between using an evaluation function and using rollouts? [1 mark]

Answer 3. The purpose of both evaluation functions and rollouts is to come up with a utility for any given state. Evaluation functions are computationally cheaper, since they need only a single pass through a function approximator such as a neural network to obtain the utility. On the other hand, the utility itself might not be sufficiently accurate/useful. With rollouts, the rollout policy is executed multiple times from the state, and the average return taken as the utility. While computationally more expensive, the advantage is that the utility is indeed an estimate (even if noisy) of the value of the state under the rollout policy.