Assignment 10 (Sol.)

Reinforcement Learning

Prof. B. Ravindran

1. Suppose that in solving a problem, we make use of state abstraction in identifying solutions to some of the sub-problems. In this approach, is it possible to obtain a recursively optimal solution to the original problem?

   (a) no

   (b) yes

   Sol. (b)
   A recursively optimal solution to a hierarchical RL problem can be obtained by combining the optimal solutions of all sub-problems. An abstraction is called safe if optimal solutions in the abstract space are also optimal in the original space. Essentially, when performing state abstraction, if we include all relevant features, then the optimal solutions in the abstract space will remain optimal in the original space. Thus, using such abstractions will allow us to obtain solutions to the original problem which are recursively optimal.

2. What kind of solution would you expect to obtain if, in solving a problem, policies for each individual sub-problem are learned in isolation (i.e., without taking into consideration the overall problem)?

   (a) hierarchically optimal solution

   (b) recursively optimal solution

   (c) flat optimal solution

   Sol. (b)
   In isolation, the best you can expect to do is solve each sub-problem optimally. Combining such policies for the sub-problems of a given hierarchical problem will generally result in a recursively optimal solution.

3. Do the policies of individual options need to be defined over the entire state space of the MDP (of the original problem)?

   (a) no

   (b) yes

Sol. (a)
If the termination condition for a particular state is equal to 1, then there is no need for the option policy to be defined over this state. Additionally, by setting the termination condition appropriately, it should be clear that the set of states which the corresponding option can be active in can be controlled and can be designed to be only a subset of the original state space. Hence, the policy of an option does not necessarily need to be defined over the entire state space.

4. Consider the two room example discussed in the lectures. Suppose you define two options, O_1, to take the agent in room 1 (the left room) to room 2, and O_2, to take the agent in room 2 to the goal state. Assuming that you have appropriately specified the initiation sets and termination conditions for both the options and are trying to learn the individual option policies, would you need to use SMDP Q-learning or would conventional Q-learning suffice?

   (a) SMDP Q-learning

   (b) conventional Q-learning

   Sol. (b)
   Since neither option invokes the other in this example, each option's policy comprises of selecting primitive actions only, and hence, SMDP Q-learning is not required.

5. Consider a Markov policy over options $\mu : S \times O \to [0, 1]$, where S is the set of states and O is the set of options. Assume that all options are Markov. While the policy μ selects options, by considering the primitive actions being selected in those options, we can determine another policy, π, which corresponds to μ, but is a conventional policy over actions. In general, will π also be a Markov policy?

   (a) no

   (b) yes

   Sol. (a)
   When considering the conventional policy, π, the action selected in state s_t depends not only on the current state (s_t), but also on the option being followed at that time, which essentially depends on the entire history since the policy μ was initiated.

6. Consider the following problem design. You have a grid world with several rooms, as discussed in the lectures, with the goal state in a corner cell of one of the rooms. You set up an agent with options for exiting each of the rooms into the other. You also allow the agent to pick from the four primitive actions. There is a step reward of -1. The learning algorithm used is SMDP Q-learning, with normal Q-learning updates for the primitive actions. You expect the agent to learn faster due to the presence of the options, but discover that it is not the case. Can you explain what might have caused this?

   (a) the options are not useful for solving the problem, hence the slowdown

   (b) initially, the agent will focus more on using primitive actions, causing slowdown

   (c) due to the presence of options, we can no longer achieve the optimal solution, hence it takes longer

   (d) it takes longer because we are using SMDP Q-learning compared to conventional Q-learning which is faster

Sol. (b)
Consider the contrast between selecting a primitive action and an option during the initial phase of learning in this problem. On selecting a primitive action from states near the start state or a doorway state, you do not reach the goal state and get a unit of negative reward. On selecting an option, however, you are transported to some doorway state (which is not the goal state) and get a negative reward of larger magnitude (recall SMDP rewards). This suggests that initially, the primitive options will appear more promising and will be explored more whereas the benefit of the options will not initially be realised or exploited. Thus, the expected speedup would in general not be observed.

7. Using intra-option learning techniques, we can learn about options even without ever executing them. True or false?

   (a) false

   (b) true

   Sol. (b)
   As we have seen, in intra-option learning, we can update estimates of multiple options based on observing the rewards received on executing a single option. Many of the options whose estimates are updated need not even have been executed.

8. Suppose that you have identified a set of sub-tasks for solving a large problem using the hierarchical learning approach. To solve each sub-task efficiently, you want to constrain the primitive actions that can be executed within each sub-task. Specifying such constraints is possible in

   (a) options

   (b) HAMs

   (c) both

   Sol. (c)
   Specifying the option policies allows us to constrain the primitive actions that are executed in each option/sub-task. Similar effect can be obtained in HAMs by limiting the choice states in each machine.