CS6700 : Reinforcement Learning
Written Assignment #3

Deadline: ??

- This is an individual assignment. Collaborations and discussions are strictly prohibited.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- **Please start early.**

AUTHOR :Manoj Bharadhwaj. S.
ROLL NUMBER :CS20Z003

1. (3 marks) Consider the problem of solving POMDPs using Deep Reinforcement Learning. Can you think of ways to modify the standard DQN architecture to ensure it can remember histories of states. Does the experience replay also need to be modified? Explain.

   Solution: We can use LSTMs after the first post-convolutional fully-connected layer. The recurrent layer, will account for arbitrarily long history of states.

   The experience replay can also be altered for better performance. Instead of the classic state, action, reward (at time t)and the next state in the experience replay. We can couple the action and observation space so that better inferences can be made.

2. (4 marks) Exploration is often ameliorated by the use of counts over the various states. For example, one could maintain a visitation count $N(s)$, for every state and use the same to generate an intrinsic reward ($r_i(s)$) for visiting that state.

   $$r_i(s) = \tau \times \frac{1}{N(s)}$$

   However, it is intractable to maintain these counts in high-dimensional spaces, since the count values will be zero for a large fraction of the states. Can you suggest a solution(s) to handle this scenario? How can you maintain an approximation of the counts for a large number of states and possibly generalize to unseen states?

Solution: True, in high-dimensional spaces, keeping track of state count will lead to problems.

- We can use static hashing to tackle this issue. i.e.

  We can define a hash function, where the states are mapped to hash codes, which allows to count their occurrences with a hash table. These counts can then be used to compute a reward bonus according to the classic count-based exploration theory.

- We can use function approximators to approximate the state space and all the model parameters with a more compact one. Then, we can continue with this count based technique.

3. (5 marks) Suppose that the MDP defined on the observation space is k-th order Markov, i.e. remembering the last k observations is enough to predict the future. Consider using a belief state based approach for solving this problem. For any starting state and initial belief, the belief distribution will localize to the right state after k updates, i.e., the true state the agent is in will have a probability of 1 and the other states will have a probability of 0. Is this statement true or false? Explain your answer.

Solution:

This statement is false.

The probability estimate of a state is a conditional probability. If the final probability has to be 1, then all other probabilities would also have to 1. Since we are choosing a starting state with any prior distribution, this isn't possible.

4. (3 marks) Q-MDPs are a technique for solving the problem of behaving in POMDPs. The behavior produced by this approximation would not be optimal. In what sense is it not optimal? Are there circumstances under which it can be optimal?

Solution:

We solve the problem assuming that we have access to the state at that point. But, the Q-MDPs use a heuristic for converting the policy into an execution policy. Thus, the policy that gets executed might not be the optimal policy for the POMDP.

We can define an MDP, where the states are the belief states and the transitions between the states is given by the computation performed to finding the belief states.

Now, we can predict the next belief state, given the present belief state and the action performed by solving this MDP. In that case, the policies can be optimal as we are trying to determine the original policy.

5. (3 marks) What are some advantages and disadvantages of A3C over DQN? What are some potential issues that can be caused by asynchronous updates in A3C?

Solution: Advantages of A3C over DQN:

- A3C is asynchronous, i.e. there are multiple agents working on the same problem. These updates make this model generalize well and also make it robust.
- Speed
- It is sample and memory efficient.

Disadvantages:

- A3C doesn't have an experience replay.
- The target network in DQN makes is very powerful.

Due to completely asynchronous updates, the A3C agents may learn same policy multiple times. Also, initialization is a challenge in this setup.

6. (6 marks) There are a variety of very efficient heuristics available for solving deterministic travelling salesman problems. We would like to take advantage of such heuristics in solving certain classes of large scale navigational problems in stochastic domains. These problems involve navigating from one well demarcated region to another. For e.g., consider the problem of delivering mail to the office rooms in a multi storey building.
   (a) (4 marks) Outline a method to achieve this, using concepts from hierarchical RL.

Solution: Here, the ultimate goal is to deliver mail to the office rooms accurately. The following are the list of things, the agent has to do to complete the process.:

1. Locating the building.
2. Navigating to the located building.
3. Locate the office rooms.
4. Navigate to the office rooms.
5. Locate the receiver.

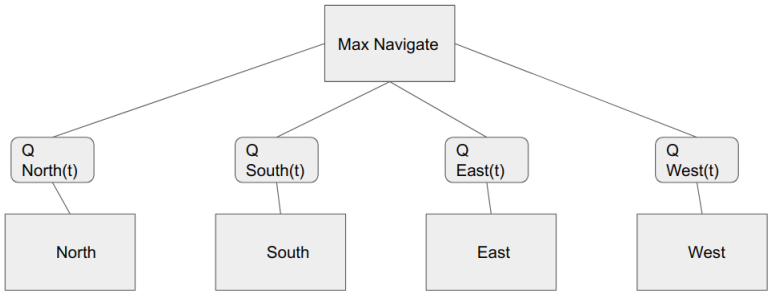
6. Select the correct mail to deliver.
7. Deliver the mail
8. Exit the room.
9. Exit the multi storey building.

Framework chosen: Max Q

MAXQ is a hierarchical learning algorithm in which the hierarchy of a task is obtained by decomposing the Q value of state-action pair into the sum of two components $Q(p, s, a) = V(a, s) + C(p, s, a)$, where $V(a, s)$ is the total expected reward received when executing the action a in state s (classic Q) and $C(p, s, a)$ is the total reward expected from the performance of the parent-task.

Here, we can solve the sub task without reference to the context in which it is executed. This creates state abstraction.

We can use this technique to train an agent to navigate to the door. The door can be the door to a building or door to a room.



Later on, the navigate itself will become an option and the agent will be able to learn the action better.

Here, each edge will have a reward associated with it and in our case the reward will be high, if the step took us near the goal (which is the door).

Followed by the navigation, the agent has two more options that is deliver/ collect the mails.

The navigate action will happen continuously in this setting, i.e inside the building, to the room and vice versa.

(b) (2 marks) What problems would such an approach encounter?

Solution: This problem with this approach is that, the recursively optimal policies can be highly suboptimal policies at times and the whole architecture is too complex.

7. (6 marks) This question may require you to refer to https://link.springer.com/content/pdf/10.1007/BF paper on average reward RL. Consider the 3 state MDP shown in Figure 1. Mention the recurrent class for each such policies. In the average reward setting, what are the corresponding ρ^{π} for each such policy ? Furthermore, which of these policies are gain optimal ?

   (a) (3 marks) What are the different deterministic uni-chain policies present ?

   Solution:
   1. $\pi(A) = a1$, $\pi(B) = a1$ and $\pi(C) = a2$
   2. $\pi(A) = a2$, $\pi(B) = a1$ and $\pi(C) = a2$
   3. $\pi(A) = a3$, $\pi(B) = a1$ and $\pi(C) = a2$
   4. $\pi(A) = a1$, $\pi(B) = a1$ and $\pi(C) = a3$
   5. $\pi(A) = a3$, $\pi(B) = a1$ and $\pi(C) = a3$

   (b) (3 marks) In the average reward setting, what are the corresponding ρ^{π} for each such policy ? Furthermore, which of these policies are gain optimal ?

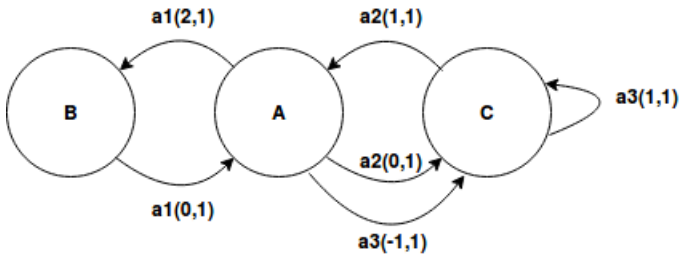


Figure 1: Notation : action(reward, transition probability). Example : a1(3, 1) refers to action a1 which results in a transition with reward +3 and probability 1

Solution: In the average reward setting, $\rho^{\pi}(x)$ associated with a particular policy π and state x is given by,

$\rho^{\pi}(x) = \lim_{n \to \infty} \frac{E(\Sigma_{t=0}^{N-1} R_t^{\pi}(x))}{N}$

Here, $R_t^{\pi}(x)$, is the reward at time t starting from state x.

1, 4 and 5 are optimal.