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 : Name.
ROLL NUMBER :

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: Modification in Replay memory: Instead of keeping only current state and observation as one sample, we can keep for the past k states.
   Modification in DQN: No we have to use those k states to get to get the q value. It can be done in many ways.

   - We can use simple autoencoder, concatenate the k states and get the compressed representation of the state and feed it to neural network to get q values.
   - We can use LSTM as well.

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: Since here we have to explore using given intrinsic reward .We can parameterize the count. We can use neural net for input state and output count. we pass the state representation and and lets it outputs o(count) we can make target as o+1, and we can find the loss $(o+1-o)^2$ or we can assume o+1 as constant, as we have to increase the value of o so we have to do gradient ascent.

Another approach is to use dynamic hashing techniques.

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 should be 1. Choosing some starting state with some 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: Since we solve the problem by assuming that you have access of state at that point of time, now we are only doing heuristic for converting it an execution policy. So policy might not be optimal Given the uncertanity in state there might have better action to pick in terms of the total reward we get.
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: DQN relied heavily on GPUs. A3C beats DQN easily, using just CPUs: A3C uses a 'forward-view' and n-step updates

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.
• DQN relied heavily on GPUs. A3C beats DQN easily, using just CPUs
• A3C uses a 'forward-view' and n-step updates

Disadvantages:
• A3C doesn't have an experience replay.
• A3C doesn't have target network.
• Due to completely asynchronous updates, the aggregated updates will not be optimal for multiple agents.

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: 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. Locate where the building is.
2. Navigating to the building location.
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.
Lets considering solving the above problem using MaxQ. Lets assume above steps as subtask.Ex- - Navigation to different level of floors, Navigating to different office rooms, Selecting the mail to be deliver.As our main task is not that much critical so if we have sub optimal policy it would be of no danger. since we know that each task is itself a smdp. we can define different options at in different task.

Navigation to different level of floors task can have options like Going to floor 6, going to floor 7 etc. Generally we use argmax and epsilon for selcting an option.

We can define heuristic function over states or over option. we can use heuristic to select between the option.We can explore with epsilon and select the options using heuristic.In this way instead of learning q values we can learn heuristics or we can combine both heuristic and q values . In this we can modify option framework to use heuristic.

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

Solution: Because it is heuristic not guaranteed to have optimal policy.
There is no way how to evaluate policy.
Defining or finding a heuristic is also a problem.
It can be too comlplex. if we use environment specific heuristics then it would not be transferable.

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

   (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 ?

Solution: Solution:
Policies are numbered according to above sequence

1. 1

2. 0.5

3. 0

4. 1

5. 1

policy 1,4,5 are gain optimal.

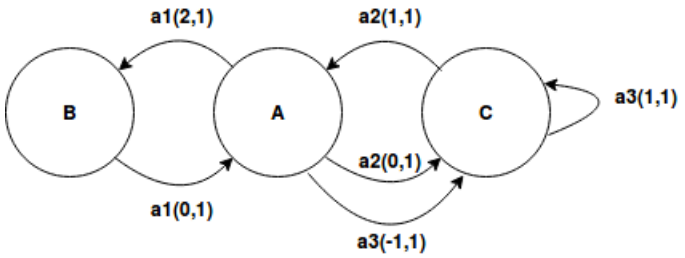

Figure 1: Notation : action(reward, transition probability). Example : a1(3, 1) refers to action a1 which results in a transition with reward +3 and probability 1