CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL **Deadline**: 30 April 2023, 11:59 pm
Name: Royyuru Sai Prasanna Gangadhar **Roll Number: ME19B190**

- 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.
- Type your solutions in the provided LaTeXtemplate file.
- **Please start early.**

1. (3 marks) Recall the four advanced value-based methods we studied in class: Double DQN, Dueling DQN, Expected SARSA. While solving some RL tasks, you encounter the problems given below. Which advanced value-based method would you use to overcome it and why? Give one or two lines of explanation for 'why'.

   (a) (1 mark) Problem 1: In most states of the environment, choice of action doesn't matter.

   Solution:
   Duelling DQN can be made use of to overcome this problem(choice of action does not matter -¿ which is equivalent to value functions being approximately equal). Dueling DQN makes use of Advantage function. This architecture leads to better policy evaluation in the presence of many similarly valued actions. The advantage function indicates how well performing a particular action is better off compared to the value of the state($A(s, a) = Q(s, a) - V^\pi(s)$).

   (b) (1 mark) Problem 2: Agent seems to be consistently picking sub-optimal actions during exploitation.

   Solution: We all are familiar with the maximization bias of DQN. This is because the DQN algorithm uses the same samples both to determine the maximizing action and to estimate its value leading to maximization bias. This maximization bias will lead to suboptimal policies. Double DQN algorithm can be used to mitigate this maximization bias. It makes use of two networks/estimators one for action selection and other for action evaluation. By doing so, we obtain an unbiased estimator for the expected Q value.

   (c) (1 mark) Problem 3: Environment is stochastic with high negative reward and low positive reward, like in cliff-walking.

   Solution: Because there is high variance in the environment the traditional SARSA suffers from high variance. Expected SARSA is a variation of SARSA which bases its update, not on $Q(s_{t+1}, a_{t+1})$, but on its expected value $E[Q(s_{t+1}, a_{t+1})]$. This expectation reduces the variance and thus in practice we can increase the learning rate in order to speed up learning. Because of lower variance, the convergence in expected SARSA will be faster compared to SARSA.

2. (4 marks) Ego-centric representations are based on an agent's current position in the world. In a sense the agent says, I don't care where I am, but I am only worried about the position of the objects in the world relative to me. You could think of the agent as being at the origin always. Comment on the suitability (advantages and disadvantages) of using an ego-centric representation in RL.

Solution: With egocentric input the position of all objects and other agents is given with respect to the position of the perceiving agent. On the other hand allocentric representation is with respect to a fixed coordinate system.

Advantages:

- It is very easy to work with egocentric representation as it is a more interpretable way to representing a state based on observability.
- It captures the local/partial surroundings information(which acts as a surrogate for the position of the agent in the environment), which can improve learning. With egocentric learning, the agent tends to develop immediately beneficial behavior more quickly

Disadvantages:

- If several parts of the environment are same, the ego centric representation falls apart to learn a reasonable policy
- There is very limited vision/scope for the agent in ego centric representation. Thus, it might not generalize to other states and environments as much as allocentric representation does.
- When we start trying to propagate reward information backwards we can get some unstable behavior and the behaviour of the agent is highly dependent on the values of gamma(discounting factor) and the environment.
- Highly vulnerable to noise as the information that agent receives is provided by the sensors installed in it. Thus, it is very important to use noise free sensors.

3. (12 marks) Santa decides that he no longer has the memory to store every good and bad deed for every child in the world. Instead, he implements a feature-based linear function approximator to determine if a child gets toys or coal. Assume for simplicity that he uses only the following few features:

- Is the child a girl? (0 for no, 1 for yes)
- Age? (real number from $0 - 12$)
- Was the child good last year? (0 for no, 1 for yes)
- Number of good deeds this year
- Number of bad deeds this year

Santa uses his function approximator to output a real number. If that number is greater than his good threshold, the child gets toys. Otherwise, the child gets coal.

(a) (4 marks) Write the full equation to calculate the value for a given child (i.e., $f(s, \vec{\theta}) = \ldots$), where s is a child's name and $\vec{\theta}$ is a weight vector $\vec{\theta} = (\theta(1), \theta(2), \ldots, \theta(5))^{\mathrm{T}}$. Assume child s is described by the features given above, and that the feature values are respectively written as ϕ_s^{girl}, ϕ_s^{age}, ϕ_s^{last}, ϕ_s^{good}, and ϕ_s^{bad}.

Solution: $f(s,\vec{\theta}) = \Theta^T.\Phi = [\theta(1),\theta(2),\theta(3),\theta(4),\theta(5)].\ [\phi_s^{\text{girl}}\ ,\phi_s^{\text{age}}\ ,\phi_s^{\text{last}}\ ,\phi_s^{\text{good}}\ ,\phi_s^{\text{bad}}\ ]^T$
$= \theta(1).\phi_s^{\text{girl}}\ + \theta(2).\phi_s^{\text{age}}\ + \theta(3).\phi_s^{\text{last}}\ + \theta(4).\phi_s^{\text{good}}\ + \theta(5).\phi_s^{\text{bad}}$

$$\text{Present given by Santa} = \begin{cases} toys, & \text{if } f(s,\vec{\theta})\ \geq threshold \\ coal, & \text{otherwise} \end{cases}$$

(b) (4 marks) What is the gradient $\left(\nabla_{\vec{\theta}} f(s,\vec{\theta})\right)$? I.e. give the vector of partial derivatives

$$\left(\frac{\partial f(s,\vec{\theta})}{\partial\theta(1)}, \frac{\partial f(s,\vec{\theta})}{\partial\theta(2)}, \cdots, \frac{\partial f(s,\vec{\theta})}{\partial\theta(n)}\right)^{\mathrm{T}}$$

based on your answer to the previous question.

Solution: Since the above gradient is linear in terms of features space, the gradient computation reduces to $\left(\nabla_{\vec{\theta}} f(s,\vec{\theta})\right) = [\phi_s^{\text{girl}}\ ,\phi_s^{\text{age}}\ ,\phi_s^{\text{last}}\ ,\phi_s^{\text{good}}\ ,\phi_s^{\text{bad}}\ ]^T$

(c) (4 marks) Using the feature names given above, describe in words something about a function that would make it impossible to represent it adequately using the above linear function approximator. Can you define a new feature in terms of the original ones that would make it linearly representable?

Solution: Let us say the original function approximator is of the form:
$= 1.\phi_s^{\text{girl}}\ + 2.\frac{1}{\phi_s^{\text{age}}\ +1} + 4.3.\phi_s^{\text{last}}\ + 3.5.\frac{\phi_s^{\text{good}}}{\phi_s^{\text{bad}}\ +0.5}$
Please not that the above equation is still a linear function approximator(linear in parameters) but the original function approximator cannot adequately represent the above function. New features that can be defined to make it linearly representable are:

$$\text{features} = \frac{1}{\phi_s^{\text{age}}\ +1}, \frac{\phi_s^{\text{good}}}{\phi_s^{\text{bad}}\ +0.5}$$

4. (5 marks) We typically assume tabula rasa learning in RL and that beyond the states and actions, you have no knowledge about the dynamics of the system. What if you had a partially specified approximate model of the world - one that tells you about the effects of the actions from certain states, i.e., the possible next states, but not the exact probabilities. Nor is the model specified for all states. How will you modify Q learning or SARSA to make effective use of the model? Specifically describe how you can reduce the number of *real* samples drawn from the *world*.

Solution: Let $S_{subset} \in S$, where S is the state space of the environment, be a subset of state space, where we have the possible next state information. Because we have possible next state information we have a rough estimate for the value function. The Q learning update can be given by $Q(s_{t+1}, a_{t+1})$:

$$Q(s_{t+1}, a_{t+1}) = \begin{cases} \frac{\sum P(\frac{s_{t+1}}{s_t, a_t}).argmax_a(Q(s_{t+1}, a))}{\sum P(\frac{s_{t+1}}{s_t, a_t})}, & \text{if } s_{t+1} \in S_{subset} \\ argmax_a(Q(s_{t+1}, a)), & \text{otherwise} \end{cases}$$

The above expression is valid if we have the information on the probability distribution of the next states. Please note that the probability distribution $P(\frac{s_{t+1}}{s_t, a_t})$ need not sum to 1 as the information is not present over the entire state space(we have no information on the exact probabilities).For SARSA, the action in next state a_{t+1} can be obtained from the policy $\pi(s_{t+1})$ in the next state. The update to value function can be given as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * (r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Model based Reinforcement approach can be used to reduce the number of real samples drawn from the world. We can use the partially specified approximate model of the world as a surrogate model for the environment. Once we have a surrogate model we can generate numerous trajectories from this model other than real experiences. We then can gradually update the model as we draw more and more real samples and thus we can get more and more accurate trajectories from the model. Now we can store all these experiences(generated and real) in a Replay buffer and use traditional Q learning to get an accurate estimate of value functions.

5. (4 marks) We discussed Q-MDPs in the class as a technique for solving the problem of behaving in POMDPs. It was mentioned that 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: In Q-MDP approach we solve the problem assuming that we would have access to the state at that point in environment. But during execution/inference we are only following an heuristic(based on observations we see) for converting that into an action . Thus the policy we execute might not be optimal for the POMDP.

$$score(a_t) = \sum b(s_t).Q(s_t, a_t)$$

, where $b(s_t)$ is the belief state. At any instant in time, we take an action that maximizes the *score* of an action given by $argmax_{a_t}(score(a_t)$. There is **no guarantee** that

$$argmax_{a_t}(score(a_t)) = argmax_{a_t}(Q(s_t, a_t))$$

. Thus the execution policy can sub optimal.

The execution policy will be optimal if and only if:

1. The sensors should be noise free and the environment should be such that each state has different observabiltiy, and belief state should uniquely determine a particular state, which is highly unlikely.

2. As the *score* of an action is acting as a proxy for $Q(s_t, a_t)$, if the maximizing action for *score* and for the the value function $Q(s_t, a_t)$ remains the same over the entire state space, we can recover an optimal policy.

6. (3 marks) This question requires you to do some additional reading. Dietterich specifies certain conditions for safe-state abstraction for the MaxQ framework. I had mentioned in class that even if we do not use the MaxQ value function decomposition, the hierarchy provided is still useful. So, which of the safe-state abstraction conditions are still necessary when we do not use value function decomposition.

Solution:

Subtask Irrelevance:

Let M_i be a subtask of MDP M. A set of state variables Y is irrelevant to sub task i if the state variables of M can be partitioned into two sets X and Y such that for any stationary abstract hierarchical policy π executed by the descendants ofM_i, the following two properties hold: (a) the state transition probability distribution $P^\pi(s', N|s, j)$ for each child action j of M_i can be factored into the product of two distributions:

$$P^\pi(x', y', N|x, y, j) = P^\pi(x', N|x, y, j).P^\pi(y', N|x, y, j)$$

Result Distribution Irrelevance (Undiscounted case):

A set of state variables Y_j is irrelevant for the result distribution of action j if, for all abstract policies π executed by M_j and its descendants in the MAXQ hierarchy, the following holds: for all pairs of states s_1 and s_2 that differ only in their values for the state variables in Y_j.

$$P^\pi(s'|s_1, j) = P^\pi(s'|s_2, j)$$

Note that this is only true in the undiscounted setting. With discounting, the result distributions are not the same because the number of steps N required depends very much on the starting location of the taxi. Hence this form of state abstraction is rarely useful for cumulative discounted reward.

If the reward function (R) is constant (which is not very true for all the environments), then even Leaf Irrelevance can hold. This condition is satisfied by the primitive actions North, South, East, and West in the taxi task, where all state variables are irrelevant because R is constant.e passenger). Hence, the taxi's initial position is irrelevant to its resulting position. So primarily, the "Subtask Irrelevance" and "Result Distribution Irrelevance" are the two safe state abstraction conditions that are still necessary when we do not use value function decomposition.

7. (4 marks) One of the goals of using options is to be able to cache away policies that caused interesting behaviors. These could be rare state transitions, or access to a new part of the state space, etc. While people have looked at generating options from frequently occurring states in a goal-directed trajectory, such an approach would not work in this case, without a lot of experience. Suggest a method to learn about interesting behaviors in the world while exploring. [*Hint: Think about pseudo rewards.*]

Solution: There are many implementation methods we can try to include pseudo rewards to learn about interesting behaviours in the world.

- **Pseudo rewards inversely proportional to frequency of update:** We can define pseudo a reward on visiting a particular state as the inverse of the number of times it visited that state. Thus, the agent prioritizes to reach the states that are visited less frequently.

$$r(s_{t+1}) = \frac{1}{freq_t + 1}$$

- **TD error**: We can also define pseudo rewards based on TD error. This pseudo reward can be added to the transition reward that the agent receives. The states that are visited less frequently will be given more and more priority because of large TD error.

One potential drawback of these methods could be that these may still require lot of experience to learn a reasonable value function over the states. A better approach could be to use model based Reinforcement learning with function approximation(for very large or continuous state space) and normal Q table(for a reasonable state space size). We can train a model as a surrogate for the environment. Using the trained model we generate lot of trajectories. We can update the Q table with the obtained trajectories. Again we use the same approach of using the absolute value of TD error as a pseudo reward given to the agent for reaching a particular state.

If we find a completely new transition:

1. There will be a high pseudo reward. Thus, the agent prioritizes to reach this particular state in the future or another approch could be to also define an option that get initiated in any part of the state space and terminates in this state where there is a high TD error. This will increase exploration, without lot of experience.

2. We need to update our model as well, as there is a good chance that the information of this completely new transition might not be captured during the initial training phase to generalize the transition appropriately.

In summary, the following can be a method to learn about interesting behaviors in the world while exploring:

1. Train a model that acts as a surrogate for the environment

2. Generate multiple trajectories using this model apart from real experiences

3. Update the Q table/function approximator based on observed and generated transitions

4. The absolute value of TD error is given as a pseudo reward to the agent on reaching a particular state. So, indirectly higher the TD error, greater the reward the agent receives, the more the agent prioritizes to reach that state. An alternate way could also be to define an option that can get initiated in any state of the environment and terminates in this particular state, where the transition exhibited a high TD error.

$$pseudo_reward(s_t) = TDerror(s_t, a_t, s_{t+1}) = |r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

, where $a_{t+1} = \pi(s_{t+1}$ for SARSA, and $argmax_a Q(s_{t+1}, a_{t+1})$ for Q learning.

5. Update model from real experiences

6. Repeat till convergence or for required number of epochs.