CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL **Deadline**: 30 April 2023, 11:59 pm
Name: K Kamalesh Kumar **Roll Number: CE20B054**

- 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:** Expected SARSA, since the bootstraped value function of the next state is found as an expectation over all actions according to the policy considered.

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

   > **Solution:** Double DQN, since this will prefer a more optimal policy over a safe policy as in the case with SARSA. The issue of maximization bias (the case where an optimal action may not be positively rewarding) will also be tackled using two different networks for finding the max action and its value, as in Double Q-Learning.

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

   > **Solution:** Dueling DQN, since it considers advantage into account (eg: the value of the state as a baseline) along with Q-values, both of which are estimated in a two-branch fashion. Hence, the excess over the advantage will not be affected much by the high negative reward (Kind of like centralizing the Q-values.)

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: Advantages: Ego-centric representations are more efficient computationally and have lower memory constraints since they'll need to represent only the immediate neighborhood of the agent and not the entire state of the environment. Thus, the complexity of the state space will significantly diminish as we can approximate many regions of the environment to be similar wrt the agent. Further, such a representation will also aid in generalizing to newer environments since only the local region around the agent matters, the agent could transfer its learning between them.

Disadvantages: The major drawback of ego-centric representations is the fact that it purely depends on the agent's relative position wrt its surroundings and its current point of view. This can be detrimental in cases where information and structure of the global state of the environment is required to pick an optimal action. Navigating through the environment to a required location will also be a problem since the agent isn't aware of its absolute location. Finally, ego-centric representations are susceptible to changes in the scale of the environment since the observable neighborhood will also get scaled. This is not the case with the allocentric counterpart and the conventional absolute reference-based representations.

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: The value of a child is given by the following linear function approximator:

   $$f(s, \vec{\theta}) = \phi_s^T \, \vec{\theta}$$

   (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: The gradient of the value is given by:

$$\nabla_{\vec{\theta}} f(s, \vec{\theta}) = \phi_s$$

(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: Any function that is not linear in nature is not representable by the above approximator. Simple example would be a function that is quadratic or polynomial in the features defined above. If the nature of the function in known apriori, then new features which resemble the feature complexity of the function can be created (for eg: a polynomial combination of the features). The resultant new features can then be linearly represented using the parameters $\vec{\theta}$.

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: Model-based reinforcement learning techniques can be used to effectively approximate a model of the environment's transition dynamics. One such framework is Dyna-Q, which apart from using the real-world experiences to learn the model, also uses them to learn the policy directly (which is also updated using samples from the model).

To reduce the number of real samples needed, the balance between real-world experience and simulated experience can be adjusted. Q-Learning or SARSA updates can be made from a combination of real and simulated experiences. If the model is accurate, we can rely more heavily on simulated experience and reduce the amount of real-world experience needed. On the other hand, if the model is inaccurate or uncertain, we may need to rely more heavily on real-world experience to ensure that the agent's policy is robust and generalizable. Further, there are many ways of enhancing exploration, such as Bayesian exploration bonus, Dyna-Q+, and intrinsically motivated exploration. Cleverly boosting exploration could aid in sample efficiency by allowing to observe more diverse experiences and not just those from greedy behavior.

Another exciting research direction that has originated recently, which also helps with reducing the no.of samples needed, is the idea of Reincarnating RL where prior computational work (e.g., learned policies, observable transition behavior) is reused or transferred to an RL agent.

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: The policy learned through the Q-MDP framework can be represented as below:

$$\arg\max_a \ score(a) = \arg\max_a \ \sum bel(s)\, Q(s,a)$$

The behavior produced by the Q-MDP algorithm can be suboptimal in the sense that it may not always choose the action that maximizes the expected cumulative reward. This is because the Q-MDP algorithm treats the belief state as a state in the MDP, which means that the algorithm does not fully account for the uncertainty in the environment. Instead, it assumes that the current belief state accurately represents the true state of the environment.

However, in some circumstances, the behavior produced by the Q-MDP algorithm can be optimal. This is particularly true in situations where the environment is sufficiently well-known that the belief state accurately represents the true state of the environment. In such cases, the Q-MDP algorithm can produce behavior that is close to optimal. However, in more complex environments with significant uncertainty, the behavior produced by the Q-MDP algorithm is likely to be suboptimal.

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: There are a total of five conditions for safe-state abstraction in the MAXQ framework specified by Dietterich. To not consider the value function decomposition but still preserve the hierarchy, the following two conditions are still necessary:

- Subtask irrelevance

  A set of variables are irrelevant to a subtask i if there exists a partition of the state variables of the original MDP such that the following is true:

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

  where x and x' give values for variables in the set X and y and y' give values for variables in the set Y

- Leaf irrelevance

  For any pair of states s_1 and s_2 that differ only in their values for the variables in Y,

  $$\sum_{s_1'} P(s_1'|s_1, a)R(s_1'|s_1, a) = \sum_{s_2'} P(s_2'|s_2, a)R(s_2'|s_2, a)$$

The remaining three conditions for state abstraction are not necessary since they deal with values of the complete function and hence are not required when value function decomposition is not considered.

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: We can define a function that assigns a pseudo-reward to each state or action based on some measure of novelty that dictates interesting behaviors. For example, we could use the entropy of the state distribution as a measure which can also be empirically approximated (through history or count frequencies), and assign a pseudo-reward to states with high entropy. Alternatively, we could use the change in entropy between successive states as a measure of interesting state transitions and assign a pseudo-reward to options that lead to large changes in entropy.

Once we have defined a pseudo-reward function, we can use it to guide exploration by incorporating it into the agent's reward signal. Specifically, we can add the pseudo-reward to the actual reward received by the agent at each time step, creating a composite reward signal that encourages both task completion and exploration of interesting behaviors. Over time, the agent will learn to associate certain states or options with high pseudo-rewards and will be more likely to visit those states or perform those actions in the future. We could potentially also vary the distribution of pseudo-rewards as the agent explores and learns so as to keep rewarding new behavior while also solving for the core reward/MDP.