CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL **Deadline**: 30 April 2023, 11:59 pm
Name: K S MOHAN KUMAR **Roll Number:ME19B118**

- 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:** Dueling DQN is a good choice. This is because Dueling DQN separates the estimation of the state value function and the state-dependent action advantage function, which allows the agent to learn the value of each state regardless of the action taken, and choose actions that are better for states where the choice of action matters.

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

   > **Solution:** Double DQN is a good choice. This is because Double DQN uses a separate network for action selection and target value estimation, which can help prevent overestimation of the action values and lead to better exploitation of the learned policy.

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

   > **Solution:** Expected SARSA is a good choice.Expected Sarsa is particularly effective in stochastic environments because it explicitly considers the expected value of the next state-action pair instead of using the actual next state and reward. It can also handle environments with a mix of positive and negative rewards by using an expected value of the action values instead of the maximum value.

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 of ego-centric representations in RL include:

1. Flexibility: Ego-centric representations can be more flexible and adaptable to changes in the environment. Since the agent's perspective is always from its own position, it can easily adjust to changes in its own position or orientation.
2. Efficiency: Ego-centric representations can be computationally more efficient than other representations, as they can encode information about the environment in a compact way.
3. Intuitive: Ego-centric representations can be more intuitive for humans to understand and interpret, as they are based on a first-person perspective.
4. Robustness: Ego-centric representations are more robust to changes in the environment, as they only depend on the relative positions of objects, rather than their absolute positions.

Disadvantages of using Ego-centric based approaches:

1. Limited perception: Ego-centric representations may not capture the full extent of the environment, as they only focus on the objects within the agent's field of view.
2. Lack of global information: Ego-centric representations may not be able to provide a complete understanding of the global state of the environment, as they only encode local information.
3. Difficulty in generalization: Ego-centric representations may not generalize well to new environments, as they are highly dependent on the agent's specific perspective and position.
4. Less Intepretability:Ego-centric representations can be less interpretable than absolute representations, as it may be difficult to understand how the agent is making decisions based on its relative positions to objects.

Ego-centric approach makes the agent to focus on immediate rewards instead of long term rewards as the agent only worries about the position of the objects relative to it.
Ego-centric approaches are widely used in autonomous driving and FPS gaming.

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

We set the decision boundary $f(s,\vec{\theta}) = C$. if $f(s,\vec{\theta}) \geq C$, the child gets the toy else the child gets the coal.

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

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

based on your answer to the previous question.

Solution: $\left(\nabla_{\vec{\theta}} f(s,\vec{\theta})\right) = \left(\phi_s^{\text{girl}}, \phi_s^{\text{age}}, \phi_s^{\text{last}}, \phi_s^{\text{good}}, \phi_s^{\text{bad}}\right)^{\text{T}}$

(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: One thing that would make it impossible to represent a function adequately using the above linear function approximator is if the function has non-linear interactions between the features. For example, if the impact of a child's age on Santa's decision making depends on whether the child is a boy or a girl, or if the effect of the number of good deeds a child has done increases quadratically of the child's age.

To make such a function linearly representable, we can define new features that capture the non-linear interactions between the original features. For example, we can define a new feature that is the product of the girl feature and the age feature, which captures the interaction between gender and age. Similarly, we can define a new feature that is the product of the square of good deeds feature and the age feature, which captures the interaction between good deeds and age. By adding such interaction terms, we can make the function linearly representable by the linear function approximator.

We could also use a kernel SVM for the data points to project the feature space into higher dimension where it will be linearly separable.

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:

If we have a partially specified approximate model of the world that tells us about the possible next states of an action from certain states, we can use this information to modify Q-learning or SARSA

to make effective use of the model and reduce the number of real samples drawn from the world. One approach to achieve this is known as model-based reinforcement learning.

In model-based reinforcement learning, the agent learns a model of the environment based on the observed data and the approximate model of the world. The agent can then use this model to simulate the future state transitions and update its value function accordingly. This approach reduces the number of real samples drawn from the world because the agent can use the model to generate simulated state transitions and update its value function based on those simulated transitions, rather than relying solely on real samples from the world.

Specifically, we can modify Q-learning or SARSA as follows:

- Learn the approximate model of the world based on the observed data and prior knowledge.
- Use the approximate model to simulate state transitions for each action in each state.
- Update the Q-values based on the simulated state transitions. This involves computing the expected value of the Q-function over the possible next states, weighted by their approximate probability of occurrence.
- Choose actions based on the updated Q-values

. This approach allows the agent to update its value function based on simulated state transitions, which can significantly reduce the number of real samples drawn from the world. Additionally, the approximate model can be used to guide exploration, by prioritizing actions that lead to more uncertain or less explored state transitions.

For example, in the Taxi-v3 environment, since we know the location of the walls of the environment, we could remove actions from the states that hit the wall or initialize negative q values at for these state action value function. This can decrease the number of real samples drawn from the world.

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 reason for this is that Q-MDPs assume that the agent has perfect knowledge of its environment, which is not the case in POMDPs. In POMDPs, the agent's observations are limited, and it has to reason about the state of the environment based on these observations. Therefore, Q-MDPs may not make the best decisions in situations where the agent's observations do not provide enough information about the environment.The optimal policy in the Q-MDP may not be optimal in the original POMDP. This is because the Q-function in the Q-MDP is an approximation of the Q-function in the POMDP, and the approximation may not be accurate in all situations.

If the agent has access to a large number of observations, or if the environment is simple enough(small state space) that the agent can make accurately estimate the belief state based on its observations, then Q-MDPs may produce optimal behavior. Additionally, Q-MDPs can be combined with other techniques such as Monte Carlo tree search to improve their performance in complex environments.

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: Safe-state abstraction is a technique used in hierarchical reinforcement learning to simplify the state space by grouping states together into abstract states that share similar properties. This simplification can make it easier for an agent to learn an optimal policy. In the MaxQ framework, safe-state abstraction is used to define subtasks and decompose them into smaller subtasks.

The conditions for safe-state abstraction in the MaxQ framework are:

1. Subtask irrelevance: Subtask irrelevance is a condition in the MAXQ framework that requires that each subtask in the hierarchy must only depend on the relevant aspects of the state space, and must be independent of irrelevant details.
2. Leaf irrelevance: The Q-values of a subtask's leaf nodes should not affect the Q-value of any max node. This means that the details of how a subtask achieves its goal should not affect the decision-making process at higher levels of the hierarchy.
3. Result distribution irrelevance: The Q-value of a max node should only depend on the expected outcome of executing its subtasks, and not on the specific way in which those outcomes are distributed across the state space.
4. Termination: Each subtask should have at least one safe termination state in the abstract state space, meaning that executing the subtask from that state will eventually lead to a terminal state.
5. Shielding: The abstract state space should shield the higher-level subtasks from the details of lower-level subtasks. This means that the abstract state space should effectively hide the details of the lower-level subtasks, allowing the higher-level subtasks to focus on their own objectives without being distracted by irrelevant details.

Even if we do not use the MaxQ value function decomposition, the hierarchy provided by the subtasks is still useful for guiding the agent's exploration and learning. Therefore, condition 1 and 2 are still necessary to maintain the hierarchy even when we do not use the 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: Pseudo-rewards are rewards that are defined by the agent rather than the environment. They can be used to incentivize the agent to explore states or actions that are not rewarded by the environment, but that the agent considers to be interesting or informative.

To use pseudo-rewards in the context of options, the agent could first perform a random walk or some other form of exploration to gather experience about the environment. The agent could then use this experience to identify interesting behaviors, such as rare state transitions or access to a new part of the state space. The agent could then define pseudo-rewards that incentivize the exploration of these interesting behaviors.

For example, the agent could define a pseudo-reward function that rewards transitions to states that are dissimilar to the states that the agent has visited before. The agent could then use this pseudo-reward function to learn options that capture the exploration of new and interesting parts of the state space. Alternatively, the agent could define a pseudo-reward function that rewards the discovery of rare state transitions, and use this to learn options that capture these behaviors.

Approaches based on using pseudo-rewards to find interesting behaviours:

- One approach is to use intrinsic curiosity-driven exploration with pseudo-rewards. In this approach, the agent generates a pseudo-reward signal based on its ability to predict the outcome of its actions. Specifically, the agent can use a model-based approach to predict the outcome of its actions and generate a pseudo-reward based on the difference between the predicted and actual outcome. This pseudo-reward signal incentivizes the agent to explore new and interesting state transitions, which can help it to discover useful behaviors that can be captured as options.
- Another approach is to use skill discovery with pseudo-rewards. In this approach, the agent generates a pseudo-reward signal based on the successful execution of a newly learned skill. The agent can use unsupervised learning techniques to discover new skills, and generate a pseudo-reward signal based on the successful execution of these skills. This approach can help the agent to discover useful behaviors that are not present in the training data.

In both cases, the key is to provide the agent with a reward signal that incentivizes exploration and discovery of interesting behaviors, while also providing guidance towards useful behaviors. Pseudo-rewards can help to achieve this by providing a way to generate reward signals for specific behaviors that may not be present in the environment or training data.