CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL **Deadline**: 30 April 2023, 11:59 pm
Name: Aaditya Kumar **Roll Number: EE21D411**

- 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 can lead to better performance as it allows for separate estimation of the state value function V(s) and the state-dependent action advantage function A(s,a). The dueling architecture can learn which states are (or are not) valuable, without having to learn the effect of each action for each state.

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

   Solution: Double DQN is preferred in this case as the agent picks sub-optimal policies due to overestimation that introduces a maximization bias in Q-learning. Double DQN involves using two separate Q-value estimators, each of which is used to update the other. Using these independent estimators, we can get unbiased Q-value estimates of the actions selected using the other estimator and thus avoid the maximization bias by disentangling our updates from biased estimates.

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

   Solution: Expected SARSA algorithm is preferred in this case as it takes into account the expected value of the next state-action pair rather than just choosing the maximum value and it is also robust to noisy rewards that makes it overcome the difficulty of stochastic environments with high negative rewards and low positive rewards, like the cliff-walking environment.

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: Ego-centric representations, where an agent perceives the environment from its own perspective, can have several advantages and disadvantages in reinforcement learning (RL):
Advantages:

- With egocentric learning, the agent tends to develop immediately beneficial behavior more quickly and will take actions optimal actions with respect to neighbouring states.
- It simplifies the agent's state space, making it easier to learn and navigate the environment as the agent is only concerned about the immediate surroundings, it need not remember the action values for far away states leading to lesser complexity in computation.

Disadvantages:

- Ego-centric representations can lead to myopic behavior, where the agent only focuses on immediate rewards and does not consider long-term consequences which may yield significantly higher rewards.
- It can be difficult for an Ego-centric agent to learn to navigate in an environment with constantly changing layouts or obstacles, as its perspective remains fixed and it may be challenging to transfer learned policies from an ego-centric representation to other representations or tasks, as the agent's learned behavior is highly dependent on the heuristics it learns about.

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 child (state) 's' is described by the following feature vector:

$$\phi(s) = \begin{bmatrix} \phi_s^{\text{girl}} \\ \phi_s^{\text{age}} \\ \phi_s^{\text{last}} \\ \phi_s^{\text{good}} \\ \phi_s^{\text{bad}} \end{bmatrix}$$

The value function can be represented by a linear combination of features weighted by a weight vector $\vec{\theta} = (\theta(1), \theta(2), \dots, \theta(5))^{\mathrm{T}}$:

$$f(s, \vec{\theta}) = \vec{\theta}^{\,\mathrm{T}} \phi(s) = \begin{bmatrix} \theta(1) & \theta(2) & \theta(3) & \theta(4) & \theta(5) \end{bmatrix} \begin{bmatrix} \phi_s^{\text{girl}} \\ \phi_s^{\text{age}} \\ \phi_s^{\text{last}} \\ \phi_s^{\text{good}} \\ \phi_s^{\text{bad}} \end{bmatrix}$$

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

The decision is taken by Santa by using the following rule, where *good_threshold_value* $\in \mathbb{R}$ and is a constant:
If the value $f(s, \vec{\theta}) \geq$ *good_threshold_value* then child 's' gets a gift;

else if $f(s, \vec{\theta}) <$ *good_threshold_value* then the child 's' gets 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

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

based on your answer to the previous question.

Solution: The gradient is given by:

$$\left(\nabla_{\vec{\theta}} f(s, \vec{\theta})\right) = \left(\frac{\partial f(s, \vec{\theta})}{\partial \theta(1)}, \frac{\partial f(s, \vec{\theta})}{\partial \theta(2)}, \cdots, \frac{\partial f(s, \vec{\theta})}{\partial \theta(n)} \right)^{\mathrm{T}} = \begin{bmatrix} \phi_s^{\text{girl}} \\ \phi_s^{\text{age}} \\ \phi_s^{\text{last}} \\ \phi_s^{\text{good}} \\ \phi_s^{\text{bad}} \end{bmatrix}$$

(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: A function that involves complex interactions between the features, such as non-linear or higher-order interactions, would be impossible to represent adequately using the above linear function approximator. For example, if the effect of age on the outcome depends on whether the child is a girl or a boy, this cannot be captured by a linear model.

One way to make the function linearly representable is to include interaction terms between the existing features. For example, we can define a new feature as follows:

$$\phi_s^{\text{girl-age-good}} = \phi_s^{\text{girl}} .\phi_s^{\text{age}} .\phi_s^{\text{good}}$$

This new feature captures the interaction between being a girl, age, and having performed good deeds. By including such interaction terms in the linear model, we can capture more complex relationships between the features and make the function linearly representable.

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: One way to modify Q learning or SARSA with a partially specified approximate model of the world is to use model-based reinforcement learning techniques. Specifically, we can use the model to simulate the possible next states and rewards, and then update our Q-values based on these simulated experiences.

To reduce the number of real samples drawn from the world, we can use a planning algorithm such as Monte Carlo Tree Search (MCTS) or Value Iteration. These algorithms use the model to simulate possible trajectories and update the Q-values based on the simulated experiences. By planning ahead and using the model to simulate possible outcomes, we can reduce the number of real samples needed to update the Q-values.

Alternatively, we can use a hybrid approach that combines model-based and model-free reinforcement learning. Specifically, we can use the model to simulate experiences in states where the model is well-specified, and use model-free reinforcement learning in states where the model is not well-specified. This approach can also help reduce the number of real samples needed to update the Q-values.

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: Solving POMDPs using Q-MDPs we consider only the MDP corresponding to the partially observable states. Policy through Q-MDP is obtained as follows:

$$score(a) = \sum_s bel(s)Q(s,a)$$

and the action that you pick is given by $\max_a Q(bel(s), a)$ where $Q(bel(s), a) = score(a)$ is value function over the belief state which can be solved as we solve regular MDPs.

The Q-MDPs solution is obtained assuming we know information about the states, but given the uncertainty in POMDP, there might have been a better action to pick in terms of the total return. Therefore, the policy obtained is optimal only for the MDP corresponding to POMDP, it might not result in an optimal policy for the entire state space i.e the POMDP as partial observability is not taken into account while obtaining the optimal policy. If we want it to be optimal, we must know all the states of POMDP with certainty.

To illustrate this we consider the scenario where there are two states, one near a cliff and the other near the goal. Despite having different goals, both states have the same representation in the framework, resulting in the same q-value. Applying the Q-MDP approach to this scenario can lead to canceling updates to the Q-value of both states, potentially producing a sub-optimal policy.

However, if the states with the same representation are symmetric, such as both being close to the cliff in the above example, then this method can become optimal. By generalizing this observation, we can conclude that if all states with the same representation are symmetric, then the Q-MDP approach can produce optimal results.

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: In the paper Dietterich puts down 5 conditions that permits safe-state abstraction out of which only two i.e. **Max Node Irrelevance** and **Leaf Irrelevance** are still necessary to maintain the hierarchy when we do not use the value function decomposition.

The rest three conditions i.e. **Result Distribution Irrelevance**, **Termination** and **Shielding** are used to remove the need of maintaining complete functions and hence these conditions are not needed in this case.

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: A simple method to learn about interesting behaviors while exploring is to use pseudo rewards, which are rewards artificially assigned to states or actions based on some criterion other than the task's true reward. Pseudo rewards can be used to encourage exploration in areas of the state space that are rarely visited, even if they do not lead directly to the task's goal.

An example of using pseudo rewards to learn about interesting behaviors that we have discussed in class is the Dyna Q+ algorithm. In this algorithm, an agent learns a model of the environment using the data collected during exploration. The model is then used to simulate additional experience, which is used to update the agent's value function. To encourage exploration of rarely visited states, the algorithm uses a pseudo reward, called the exploration bonus, based on the uncertainty of the value estimate for each state. States with high uncertainty are given higher pseudo rewards, which encourages the agent to explore those states further. By exploring these states, the agent can discover new and interesting behaviors that it may not have discovered otherwise.

Suppose an agent is navigating a maze in a grid-world problem to reach a goal state. To encourage exploration and discover new interesting behaviors, it uses the Dyna-Q+ algorithm with pseudo rewards. It assigns a high pseudo rewards to rarely visited states with high uncertainty in their value estimates. The agent can discover a hidden path that leads to a valuable object or a shortcut that reduces the time required to reach the goal state. By caching policies that lead to these behaviors, the agent can improve its performance in the task.