CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL **Deadline**: 30 April 2023, 11:59 pm
Name: Argha Boksi **Roll Number:** CS21D407

- 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:
   In a situation where the choice of action doesn't matter in most states of the environment, the most appropriate advanced value-based method to use would be Dueling DQN.
   Dueling DQN is designed to estimate the value of each state separately from the advantage of each action in that state. In situations where the choice of action doesn't matter, the advantage of each action would be close to zero, and the value of the state would be the dominant factor in determining the Q-value of that state-action pair. By separating the estimation of the value of a state from the advantage of each action in that state, Dueling DQN can estimate the value of each state more accurately and efficiently, even in situations where the choice of action doesn't matter.

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

   Solution:
   If the agent seems to be consistently picking sub-optimal actions during exploitation, the most appropriate advanced value-based method to use would be Double DQN.
   The issue of consistently picking sub-optimal actions during exploitation is often caused by overestimation of Q-values, especially in situations where the agent has not explored the environment sufficiently to accurately estimate the true values of state-action pairs. Double DQN addresses the issue of overestimation of Q-values by using two separate networks, one to select the best action and another to estimate the Q-value of that action. By decoupling the action selection and Q-value estimation, Double DQN is able to reduce overestimation and improve the accuracy of Q-value estimation.

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

Solution:

If the environment is stochastic with high negative reward and low positive reward, the most appropriate advanced value-based method to use would be Expected SARSA.

Expected SARSA is a variant of the SARSA algorithm that estimates the expected value of the next state-action pair, rather than just the maximum value. In a stochastic environment with high negative reward and low positive reward, the choice of the next action can be uncertain and the agent may need to explore different actions to avoid the high negative reward. Expected SARSA takes into account the uncertainty in the choice of the next action by estimating the expected value of the next state-action pair, leading to more accurate and robust Q-value estimation. This can help the agent to explore the environment more effectively and learn a better policy.

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 using an ego-centric representation in RL:

1. **Reduced state space:** By using an ego-centric representation, the agent can focus on the parts of the environment that are most relevant to its current goals, and ignore irrelevant information. This can significantly reduce the dimensionality of the state space and make it easier for the agent to learn an optimal policy.
2. **Faster learning:** Since the agent only needs to observe a subset of the environment, it can make faster and more accurate predictions about the effects of its actions, which can speed up the learning process.
3. **Robustness:** Ego-centric representations are often more robust to changes in the environment than other types of representations, since the agent only needs to observe local information that is likely to be stable over time.

Disadvantages of using an ego-centric representation in RL:

1. **Limited information:** Ego-centric representations can limit the amount of information available to the agent, which can make it harder for the agent to learn an optimal policy in complex environments.
2. **Sensitivity to initial position:** Ego-centric representations can be highly sensitive to the agent's initial position in the environment, which can lead to suboptimal policies if the agent starts in a position that is not representative of the overall environment.
3. **Limited transferability:** Ego-centric representations may not be easily transferable to new tasks or environments, since they are often specific to the current task and rely on knowledge of the agent's current position in the world.

Overall, ego-centric representations can be a powerful tool for reducing the complexity of the state space in RL, but they also have limitations that need to be carefully considered in the design of RL algorithms.

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:
   So here weight vector $= \vec{\theta} = (\theta(1), \theta(2), \ldots, \theta(5))^{\mathrm{T}}$
   All features are represented as - ϕ_s^{girl} , ϕ_s^{age} , ϕ_s^{last} , ϕ_s^{good} , and ϕ_s^{bad}
   We can define $f(s, \vec{\theta}) = \theta(1).f_1(\phi_s^{\text{girl}}) + \theta(2).f_2(\phi_s^{\text{age}}) + \theta(3).f_3(\phi_s^{\text{last}}) + \theta(4).f_4(\phi_s^{\text{good}}) + \theta(5).f_5(\phi_s^{\text{bad}})$
   The $f_i()$ functions can be linear or non-linear. If we use a linear function where $f_i(x) = x_i$ then we have,
   $f(s, \vec{\theta}) = \theta(1).\phi_s^{\text{girl}} + \theta(2).\phi_s^{\text{age}} + \theta(3).\phi_s^{\text{last}} + \theta(4).\phi_s^{\text{good}} + \theta(5).\phi_s^{\text{bad}}$

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

   (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: Lets suppose if santa decides to gift to kids whose age is less and who have done lesser bad deeds and if he thresholds based on functions like (square of the age+ square of number of bad deeds) (or) product of age and number of bad deeds. The above decision boundaries cannot be represented by a linear approximator.

To get around this we could add new features like square of age, square of no. of bad deeds,good deeds for the first case. Likewise we could add product of age and number of bad deeds as a feature. In this new feature space the decision boundary will become linear .

Also as a general solution we could use kernels in SVM, where we use kernel trick to cast the input feature space into higher dimensions(infinite dimensions) where they become 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 effects of actions from certain states, we can use a model-based RL approach to modify Q-learning(or SARSA) and make effective use of the model.

One approach to incorporating the model in Q-learning is to use a sampling-based algorithm called **Model-Based Interval Estimation with Exploration Bonus(MBIE-EB)**. MBIE-EB is an algorithm that uses the approximate model to compute confidence intervals on the Q-values, which are used to guide exploration and exploitation. The confidence intervals are based on the approximate model's estimate of the possible next states and rewards. Paper Link

To reduce the number of real samples drawn from the world, MBIE-EB uses the approximate model to generate simulated transitions, which are used to update the Q-values using the Bellman equation. However, because the approximate model does not provide exact probabilities, the algorithm uses a conservative estimate of the transition probabilities in the Bellman equation. This ensures that the Q-values are not over-optimistic and that the algorithm still converges to the optimal policy.

To further reduce the number of real samples drawn from the world, MBIE-EB uses value iteration. Value iteration uses the approximate model to iteratively update the Q-values until convergence, rather than relying on real samples from the environment. This approach allows the algorithm to learn an effective policy using only the approximate model, and reduces the number of real samples required for learning.

In summary, if we have a partially specified approximate model of the world, we can modify Q-learning(or SARSA) using a sampling-based algorithm called MBIE-EB. MBIE-EB uses the approximate model to compute confidence intervals on the Q-values, which are used to guide exploration and exploitation. To reduce the number of real samples drawn from the world, the algorithm uses the approximate model to generate simulated transitions, and uses value iteration to update the Q-values until convergence. This approach allows us to learn an effective policy while minimizing the number of real samples required for learning.

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?

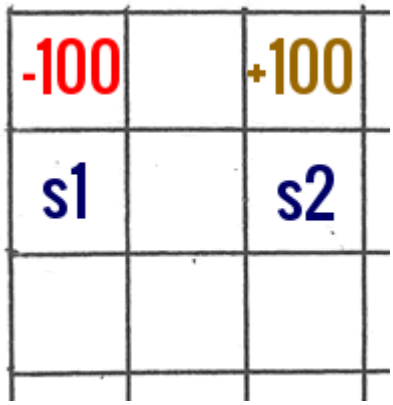
Solution:

POMDPs have partially observed states and hence the agent doesn't have the complete information about the state. So, it is tougher to solve when compared to solving an MDP.

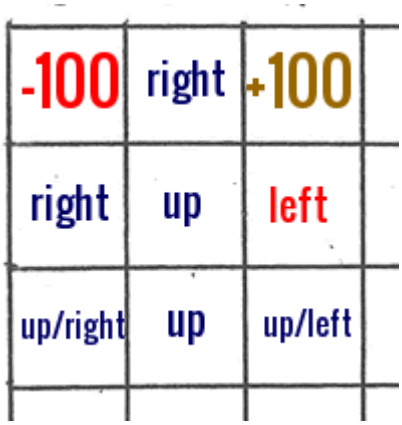
Q-MDP is one way to solve a POMDP. Here you assume that you have complete information about the MDP i.e you assume the minimal information about the state as the complete information and learn action-value or value functions. When you are executing the policy, you use some heuristic to choose the action based on the belief state for that particular state.

Consider 2 states s_1 and s_2 which have the same representation as they are partially observed. Lets call this representation f. Now, $Q(s_1, a)$ and $Q(s_2, a)$ will be learnt from the samples where state is observed as f. This means, if the agent is at s_1 and makes a transition after taking action a to some other state s_x that is used to update $Q(s_1, a)$. But since we have partial information about the state space, that transition updates $Q(f, a)$. Similarly, the transitions from state s_2 after taking action a will update $Q(f, a)$. In other words, a Q function is shared between 2 different states which have identical representations when observed partially.

Lets assume the case that, s_1 is close to a state which gives a high negative reward and s_2 is close to a state which gives a high positive reward. Now, when trained $Q(f, a)$ will have updates which partially get cancelled due to the positive and negative rewards. The gridworld is shown below.



If the agent reaches the top-right most square it gets a positive reward of +100 and if it lands up in the top-left most square it gets a negative reward of -100. States s_1 and s_2 have the same representation f. As a result $Q(f, a)$ won't be learned properly. The reason being, $Q(f, UP)$ gets cancelling updates as both the transitions s_1-UP and s_2-UP update $Q(f, UP)$. So, for state s_1 and s_2 the UP action won't be optimal. In fact if the other states have unique representations (even though they are partially observed), the policy learnt won't be optimal. The policy learnt in this case is shown below.



Now, the policy for the state s_2 is not optimal. This is because we have solved the POMDP in a Q-MDP fashion which doesn't result in optimal policies in all cases.

The policies computed using Q-MDP method will be optimal if the states which have similar representations in the partially observed space are similarly rewarding. In that case, the updates made to $Q(f, a)$ are more related and hence the policy learnt will be optimal.

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 has enlisted 5 conditions for safe state abstraction out of which only two i.e. **Subtask Irrelevance** and **Leaf Irrelevance** are still necessary to maintain the hierarchy when we do not use the value function decomposition.

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

Reference : State Abstraction in MAXQ Hierarchical Reinforcement Learning

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:

One method to learn about interesting behaviors in the world while exploring using options and pseudo-rewards is the **Option-Critic Architecture**.

The Option-Critic Architecture is a reinforcement learning framework that combines the idea of options with actor-critic methods. The basic idea is to learn a set of options, each of which is a sub-policy that can be executed for a fixed number of steps or until a goal state is reached. The

options can be learned in a goal-directed manner, but can also be learned in a non-goal-directed manner by using pseudo-rewards.

To learn about interesting behaviors in the world while exploring, the agent can be programmed to generate additional pseudo-rewards when it encounters states or actions that are related to the behaviors of interest. These additional pseudo-rewards can be used to learn new options that are specifically designed to achieve these interesting behaviors.

For example, suppose the interesting behavior is to reach a certain score or level in a game environment. The agent can generate pseudo-rewards whenever it reaches states that are closer to the desired score or level. The agent can then learn new options that specifically focus on achieving these interesting behaviors. These options can be cached away and reused in future exploration, making it more efficient to achieve these behaviors again.

Overall, using the Option-Critic Architecture with additional pseudo-rewards can be an effective way to learn about interesting behaviors in the world while exploring, even when these behaviors are rare or difficult to achieve using traditional reinforcement learning approaches. By caching away policies that caused interesting behaviors, the agent can accelerate learning and make more efficient use of its exploration time.

Reference : The Option-Critic Architecture