CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, POMDP, HRL — **Deadline**: 30 April 2023, 11:59 pm
Name: GAUTHAM GOVIND A — **Roll Number: EE19B022**

- 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
   Dueling DQN explicitly separates the representation of state Value functions ($V(s)$) and state-dependent Action Advantages ($A(s,a)$) via two separate streams. This architectural modification allows this strategy to identify states which are "valuable", in the sense the states for which the choice of action actually matters without concerning itself with the other states.

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

   Solution:
   Double DQN
   A consistent choice of sub-optimal actions during exploitation indicates that the q-values learnt by the agent are incorrect. A probable reason for this is the overestimation of q-values during the learning phase which can be corrected for by making use of Double DQN, as Double DQN was designed specifically for this purpose.

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

Solution:

Expected SARSA

When the environment is stochastic, updating using the q-value of a single action as in the case of SARSA may not be optimal as the stochasticity may introduce a high variance, leading to incorrect estimates. This high variance is corrected for by Expected SARSA, thereby leading to better performance.

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:

- **Compact state representation:** Ego-centric representation is based on the immediate surroundings of the RL agent. The agent considers each distinct surrounding setup as a different state. The number of possible setups of immediate surroundings could be much lower than the total number of states, especially in a large state space leading to possibly more compact state representation.
- **Better short-term performance:** Since ego-centric representations work with immediate surroundings, it is easier to react to immediate positive and negative rewards. The presence of a high positive/negative reward will be easier to identify in an ego-centric setup as the agent learns to identify common patterns in its immediate surroundings, as opposed to a global map in which such patterns are much more difficult to learn.
- **More robust to dynamic environments:** In cases where the world dynamics may change frequently, a global map approach may be infeasible as it might not be possible to keep track of all the changes in the internal map of the agent. Ego-centric representations are more robust in such scenarios as they do not rely on a gloabl map, but rather on patterns in immediate surroundings.

Disadvantages:

- **Poorer long-term performance:** Since the agent is only concerned with its immediate surroundings, maximizing long term rewards will be inherently difficult for ego-centric representations. Furthermore, propagation of rewards for learning methods such as Q-learning is not all obvious. It has also been observed that ego-centric representations become non-convergent for high γ values.
- **Ignorance of global positions:** Ultimately, the goal of any RL agent is to move towards a prescribed goal state. While ego-centric representations can help the agent navigate its immediate surroundings, and possibly longer distances in simple environments, a global map will be necessary to navigate to a goal state in very complex environments, which is not provided by ego-centric 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:
Since it is given that the function approximator is linear, the value for a given child should take the form:

$$f(s, \vec{\theta}) = \vec{\phi}_s \cdot \vec{\theta}$$

where $\vec{\phi}_s$ is the feature vector for child s and $\vec{\theta}$ is the vector of model weights.
Given,

$$\vec{\theta} = (\theta(1)\ \theta(2)\ \theta(3)\ \theta(4)\ \theta(5))^{\mathrm{T}}$$

and

$$\vec{\phi}_s = (\phi_s^{\text{girl}}\ \ \phi_s^{\text{age}}\ \ \phi_s^{\text{last}}\ \ \phi_s^{\text{good}}\ \ \phi_s^{\text{bad}})^{\mathrm{T}}$$

By taking dot product, we obtain:

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

If M is the value threshold used by Santa to decide whether to give gifts or not (*good* theresh-old), then child gets toys if $f(s, \vec{\theta}) > M$ and child gets coal if $f(s, \vec{\theta}) \leq M$.

(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:
From part (a), it is clear that the vector of partial derivatives is equal to the feature vector $\vec{\phi}$ since $f(s, \vec{\theta}) = \vec{\phi}_s \cdot \vec{\theta}$.
Therefore, the vector of paprtial derivatives is given by:

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

(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:

Suppose Santa now realizes that number of deeds done by each child in a year are different and hence it does not really make sense to decide based on the raw count of good deeds alone. Rather, Santa would now like to use the **fraction of total deeds that are good** to decide whether to give gifts or not. If the fraction is greater than 0.5, toys will be given, else coal will be given.

That is, the decision is now based on the value of the fraction: $\frac{\phi_s^{\text{good}}}{\phi_s^{\text{good}} + \phi_s^{\text{bad}}}$

We will not be able to compute the above fraction using the linear function approximator with feature $\vec{\phi}$. This is because division is essentially a non-linear operation.

However, suppose we were to introduce the above fraction as a new feature. Now clearly the new feature $\phi_{\text{good fraction}}$ can be written as:

$$\phi_{\text{good fraction}} = \frac{\phi_s^{\text{good}}}{\phi_s^{\text{good}} + \phi_s^{\text{bad}}}$$

If we use the new set of features, which includes $\phi_{\text{good fraction}}$, we can write the decision as:

$$\phi_{\text{good fraction}} > 0.5$$

which is a linear representation. In general, products and powers of existing features can be used as new features so as to allow for non-linearities even while using a linear function approximation.

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:

We have an approximate model of the world which specifies the next states on picking actions from certain states; however we do not have access to the exact probability distribution. If we did have access to the exact probabilities, we could have used this information to develop a Partially Known Markov Decision Process (PKMDP) model as is introduced in this paper.

But since we do not have the probability distribution, this is not possible. However, we still can look at the possible states that result from picking an action and use this information to weed out actions which only lead to bad states. Specifically, we can take the **set of actions which only lead to bad states and remove them from the possible set of actions in that state** so that these are not explored during learning. This would reduce the size of the search space and hence reduce the number of real samples drawn from the world.

Another possible approach is to make use of **artificial samples generated using the given**

approximate model of the world. As an example, consider the Q-learning update rule:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Usually, we sample the environment to obtain (s_{t+1}, r_{t+1}). Instead, now you could **artificially sample from the list of possible next states (and rewards) instead of sampling the environment directly.** This would again help reduce the number of real samples drawn from the world as these artificial samples can help q-learning/sarsa converge faster.

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:

Q-MDPs attempt to solve POMDP problems by assuming that the underlying MDP is known, however the current state is unknown. The algorithm computes q-values for the MDP, and takes a weighted average of the q-values for an action(a) across all states, weighted by the belief ($b(s)$) of being in that state, essentially $\sum_i b(s_i)Q(s_i, a)$.

However, a key assumption in the algorithm is that **one-step control leads to full observability.** The behaviour produced by Q-MDP would not be optimal due to this approximation. In many cases, POMDPs do not become fully observable leading to sub-optimal results. To see this, suppose we have an environment where a state with a very low reward and a state with a very high reward have the same representation. **Q-MDP works by assuming the same representation for both and hence both states will get the same update.** However, this is clearly sub-optimal as the bad state should receive a negative update whereas the good state should receive a positive update.

However, if it so happens that the **states with same representation are identical to the agent as far as the end goal is concerned,** then Q-MDPs may be optimal. Consider for instance a 3x1 gridworld, where the center state is the goal state. Suppose the only observable we have, o, is 1 if we have a wall to the left/ right and 0 if not. Clearly, $o = 0$ for the goal state and $o = 1$ for the other two states. Suppose further that the actions available to us are to move away from the wall and to move towards the wall. In this case, irrespective of whether the agent is actually to the left or right of the goal state, the optimal action is to move away from the wall, whenever $o = 1$. Clearly, even though we do not have full observability, the actions are optimal. In scenarios where such **symmetry** exist, Q-MDP can 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:

Diettrich specifies five conditions for safe-state abstraction in this paper. The conditions are:

1. Max Node Irrelevance
2. Leaf Node Irrelevance
3. Result Distribution Irrelevance

4. Termination
5. Shielding

Out of these 5 conditions, the first two, **Max Node Irrelevance and Leaf Node Irrelevance**, are necessary for the abstraction framework itself. This is because these are the two conditions which impose structure on the problem so as to enable state abstraction. Hence, these two conditions are still necessary.

The three remaining conditions, Result Distribution Irrelevance, Termination and Shielding essentially help define an abstract completion function (C), which is necessary for performing value function decomposition. However, if value function decomposition is not performed, this formalism is not essential. Hence, these 3 conditions need not hold if our aim is to only maintain the abstraction structure without 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:

The key idea here is to provide some sort of incentive to the agent to explore state-action pairs which have not been explored in a while. This is because if the environment is stochastic, it may happen that a state-action pair which previously gave poor rewards would give better rewards at a later stage.

This incentive can be provided by making use of pseudo rewards. One strategy would be to **provide pseudo rewards** whenever the agent exhibits "interesting" behaviour, such as a rare state transition yielding high rewards or access to a new part of the state-space. However, this would be possible only if we can engineer beneficial pseudo-rewards based on our understanding of the problem. That is, the **pseudo rewards themselves must be engineered by the designer.** While this can yield good results wherever applicable, it may not always be possible for the designer to know what pseudo rewards should be given when to achieve optimal performance.

One other alternative is to modify the rewards using some parameter available to the agent than defining a pseudo reward directly. **Time steps from the time a state-action pair was last selected** is a good indicator of how rare that particular state-action pair is. This is precisely what is leveraged by the **Dyna Q+** algorithm. The algorithm essentially adds the term $k\sqrt{\tau}$ to each reward, where k is a small positive constant and τ is the number of time steps from the last selection of the state-action pair. This allows the algorithm to select state-action pairs which have not been explored in a long time to see if the behaviour has changed to something interesting.