CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, FA, PG, AC, POMDP, HRL **Deadline**: 26 April 2022, 11:59 pm
Name: Rongali Rohith **Roll Number:** EE19B114

- 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. (4 marks) Recall the four advanced value-based methods we studied in class: PER, 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, estimates using both advantage A(s,a) and Value V(s). Explicitly separating two estimators, the dueling architecture can learn which states are valuable without having to learn the effect of each action for each state.

   (b) (1 mark) Problem 2: Sparse rewards.

   > **Solution:** Prioritised Experience Replay. It would give priority to the important states(ones with non-zero reward) to converge faster.

   (c) (1 mark) Problem 3: Agent seems to be consistently picking sub-optimal actions during exploitation.

   > **Solution:** Double DQN is preferred in this case. The model picks sub-optimal policies due to over-estimation in q-learning. This paper shows that this can be avoided using double q-learning.

   (d) (1 mark) Problem 4: Environment is stochastic with high negative reward and low positive reward, like in cliff-walking.

   > **Solution:** Expected Sarsa is preferred in this case. Stochastic environment have high variance which is countered by this method, further it improves upon the advantages of sarsa(like in the cliff walking example).

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: In egocentric representation we will have lesser number of states as it is only based on what it sees immediately around it, which would help us reduce computation. Not only that, such a representation would quickly learn to avoid taking a step when it is close to states with highly negative rewards(i.e. falling off a cliff) and quickly learn to take a step towards the goal(positive reward) when it is near such goal states.

While with these advantages, we must also note that learning in ego-centric representations mainly focuses on the immediate reward instead of the long-term reward. It becomes non-convergent when we try to impose high gamma values.

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: Linear function approximator:
   $f(s, \vec{\theta}) = (\theta(1), \theta(2), \theta(3), \theta(4), \theta(5))^{\mathrm{T}}.(\phi_s^{\text{girl}}, \phi_s^{\text{age}}, \phi_s^{\text{last}}, \phi_s^{\text{good}}, \phi_s^{\text{bad}})$
   $f(s, \vec{\theta}) = \theta(1).\phi_s^{\text{girl}} + \theta(2).\phi_s^{\text{age}} + \theta(3).\phi_s^{\text{last}} + \theta(4).\phi_s^{\text{good}} + \theta(5).\phi_s^{\text{bad}}$
   The decision boundary will be $f(s, \vec{\theta}) = 0$. if $f(s, \vec{\theta}) > 0$ child gets the gift $f(s, \vec{\theta}) < 0$ does not get the gift

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

(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: 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. (6 marks) Recent advances in computational learning theory, have led to the development of very powerful classification engines. One way to take advantage of these classifiers is to turn the reinforcement learning problem into a classification problem. Here the policy is treated as a labeling on the states and a suitable classifier is trained to learn the labels from a few samples. Once the policy is adequately represented, it can be then used in a policy evaluation stage. Can this method be considered a policy gradient method? Justify your answer. Describe a complete method that generates appropriate targets for the classifier.

Solution: Using this paper as reference.

Firstly, a brief outline of the method to generate training examples. We consider several roll-outs to get approximate q values represented by $Q(S, a)$.

After which we generate the following examples:

Positive examples : We take the statistically significant actions i.e. for a particular state, action which has much higher value than all other actions.

Negative examples :Consider the bad actions i.e. the actions whose is much lower than the optimal action for that state.

This examples are fed into a classifier(SVM or Neural network) for training. Each state-action pair will be represented by input features and the classifier would be able to classify every such pair as positive or negative i.e. if it is an optimal action for that state.

Yes the method can be considered as a policy gradient method as we are using a parametrised representation of the optimal action/policy and the parameters we learn from the positive and negative training examples that we've generated previously.

5. (5 marks) Suppose that the system that you are trying to learn about (estimation or control) is not perfectly Markov. Comment on the suitability of using different solution approaches for such a task,

namely, Temporal Difference learning, Monte Carlo methods, and Policy Gradient algorithms. Explicitly state any assumptions that you are making.

Solution:

Temporal Difference learning: It would not work for non-markovian systems. For these systems the transition probabilities and rewards depend on the history of states that we have come through. Considering an example of single-step TD, for updating value function of a state-action pair for various samples of state sequences gives different target every time and hence we will not be able to estimate correct value function.

Monte Carlo methods: It is robust to non-Markovian sysytems as well.It would simply sample sequence of states, actions and rewards, and take average of rewards for each action. As long as there is well defined rewards and we perform the tasks episodically(need not necessarily terminate) we are good to go, there is no underlying Markov assumption

Policy Gradient algorithms: It would not work for non-Markovian systems. In the proof of policy gradient theorem itself we invoke the Markovian condition.

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a) \big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

6. (2 marks) We discussed two different motivations for actor-critic algorithms: the original motivation was as an extension of reinforcement comparison, and the modern motivation is as a variance reduction mechanism for policy gradient algorithms. Why is the original version of actor-critic not a policy gradient method?

Solution:

In the original motivation which was an extension of reinforcement comparision, the values of the preferences are updated by the following equation:

$p(s_t, a_t) = p(s_t, a_t) + \beta(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$

And the action-selection probabilities are obtained by taking softmax on the above values.

In the modern motivation of AC we use gradient of expected return to update the parameters(defining characteristic of a policy gradient method).
But note that in the original motivation the values of the preferences are updated by simply using TD error without using any parametrised representations/gradients, therefore we do not consider it as a policy gradient method.

7. (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: Consider the taxi example in PA-3, where assume we have knowledge about the walls which would tell us about the actions which would ram the taxi onto the walls, we could incorporate this info by either removing such actions in those states(adjacent to walls) or by initialising those state-action pairs with highly negative values to start with. Likewise we could do the same with states near the goal state, initialise such actions with highly positive values. Without this initialisation the agent would take considerable samples to learn the same things we have incorporated initially in the q-table, hence we are cutting down on the number of samples in this way.

If we have information about the dynamics between adjacent states, in some cases both their value function can be updated together and again reducing the need for further sampling.

8. (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: While solving POMDP's using Q-MDP's we consider only the partially observable states. Though Q-MDP solves for the optimal solution with whatever input we have provided to it, it is not taking the partial observability into consideration, so it might not result in an optimal policy for the entire state space.

Lets consider two states one is near a cliff and the other is near the goal, but lets suppose both of them have the same representation in this framework hence will have same q-value, now while applying Q-MDP, there will be cancelling updates to the q-value of both the states and we might end up with a sub-optimal policy.

But this method will work if the two states in the above example with same representation are similar/symmetric in terms of their position (i.e. both are close to cliff).Generalising this we can say if all the states with same representation are symmetric, this method can become optimal.

9. (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 Ditterich 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.

Used this paper as reference

10. (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: Much like in Dyna Q+ that we have discussed in class we could assign pseudo-rewards to rare action sequences.The reward will be more with the time it was last visited. The longer it takes to revisit, the higher the exploration bonus. In this way we would be able to identify rare sequences.

For example in a gridworld problem there is a short route to the goal but it is surrounded by pits(that give high negative reward), definitely this route will be taken very rarely though its the optimal one, our method would give high pseudo reward to such action sequences and help us identify them.