CS6700 : Reinforcement Learning
Written Assignment #2

Topics: Adv. Value-based methods, FA, PG, AC, POMDP, HRL **Deadline**: 26 April 2022, 11:59 pm
Name: Vishal Rishi MK **Roll Number:** CH18B013

- 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:** The motivation for coming up with Dueling DQN is that it takes advantage of the fact that we do not need to compute the action-value for each action in many states. Since the choice of action does not matter in many states, the agent does not need to spend effort to estimate those action-values accurately. Hence, Dueling DQN would be suitable to handle this problem.

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

   > **Solution:** Since the rewards are sparse, only very few transitions would yield a non-zero reward. During learning, we need to prioritize those transitions because they provide better updates to the action-values. Hence, Prioritized Experience Replay (PER) would be suitable to overcome this problem.

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

   > **Solution:** Sometimes, during Q-learning, the agent might suffer from maximisation bias. This arises in Q-learning when we take the maximum of an estimate instead of using an estimate of the maximum. This bias might cause the learning of sub-optimal action-values which would yield a sub-optimal policy during exploitation. Hence, Double DQN would be suitable to overcome this problem.

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

Solution: IN an environment with a very high negative reward and a low positive reward, the variance of the updates might be higher than expected. Expected SARSA would be suitable to overcome this problem. Expected SARSA allows us to perform updates with less variance. This allows us to use higher learning rates and thus achieves faster convergence.

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: If the agent is performing egocentric learning, then it distinguishes situations based on it's immediately surrounding environment.In egocentric learning, each situation is determined by what is near the agent, and so new action values for each of the possible movements have to be learned whenever the immediate surrounding is not something the agent has seen before. Even if the agent is in different states of the MDP, egocentric learning might associate the same state-action values for these states, if they have the same surrounding. Because of this, the agent tends to develop immediately beneficial actions more quickly. One problem with egocentric learning is that, during learning, we might often end up on different states with the same surroundings. Because of this, propagating reward information backwards might cause unstable behaviour. Also, since the agent only has information about its surroundings, the agent might be short-sighted and greedy. This might not be suitable if we want a policy that should fetch rewards in the long run.

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 function approximator is given by,

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

   (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 of the function approximator with respect to the parameters is given by,

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

(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: The above function $f(s, \vec{\theta})$ is linear in the parameters $\vec{\theta}$. Any function that is linear in the parameters can be represented by a linear function approximator. We can either try adding different variables or transform the existing variables to get the correct linear functional form. But, if the function is non-linear in parameters, it is hard to represent it adequately using the above linear function approximator. An example would be:

$f(s, \vec{\theta}) \;=\; \phi_s^{girl^{\theta(1)}}.\phi_s^{age^{\theta(2)}}.\phi_s^{last^{\theta(3)}}.\phi_s^{good^{\theta(4)}}.\phi_s^{bad^{\theta(5)}}$
To make use of a linear functional approximator in this case, we need to define new features in terms of the original ones. The above equation can be written as:

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

Hence the new features are the logarithms of the original ones. Though logarithm is a non-linear transformation, the resulting equation is still linear in the parameters.

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: Yes, this method can be treated as a policy gradient method, if we follow the steps mentioned below. For simplicity, we assume that only two actions are available in each state. Hence, we formulate a binary classification problem.
a) First, we let the policy network π_θ play the environment several times, and at each step, we compute the gradient of the loss function (binary cross-entropy) with the chosen action (target) and the probability of choosing the action (according to the policy network) as inputs to the loss function.
b) After running several episodes, we compute the discounted return at each step. We multiply the gradients with the discounted return at each time step.
c) Finally, we compute the mean of the resulting vectors, across all the time steps. We then perform a gradient descent step.
d) The steps a), b), and c) are performed until we find a good policy.
We note the fact that the gradient of the binary cross-entropy loss function with the chosen action

as the target is equal to $\nabla_\theta ln(\pi_\theta(a_t|s_t))$. Because of this, the classification approach given above is equivalent to the policy gradients method. We can easily extend this approach to a multi-class setting (more than two actions).

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. 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: Let us assume that the under-lying process is an n-step Markov decision process. Hence the action-value function and the optimal policy in an n-step Markov decision process does not depend only on the current state. Since the Bellman equation and the policy gradient theorem assume that the decision process is Markov (the decision process depends only on the current state), if we still apply the learning methods like Temporal difference learning, Monte Carlo learning, and Policy gradients to the original decision process, we might get sub-optimal performance. Greater the dependence on the past states, poorer would be the performance of the learning methods. By concatenating n states in the original process into a single state, we create a new decision process that is Markov. Since the new process is an MDP, all the above learning methods become suitable.

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: Although the REINFORCE-with-baseline method learns both a policy and a state-value function, we do not consider it to be an actor–critic method because its state-value function is used only as a baseline, not as a critic. That is, it is not used for bootstrapping (updating the value estimate for a state from the estimated values of subsequent states), but only as a baseline for the state whose estimate is being updated. This is a useful distinction, for only through bootstrapping do we introduce bias and an asymptotic dependence on the quality of the function approximation. As we have seen, the bias introduced through bootstrapping and reliance on the state representation is often beneficial because it reduces variance and accelerates learning. REINFORCE with baseline is unbiased and will converge asymptotically to a local minimum, but like all Monte Carlo methods it tends to learn slowly (produce estimates of high variance) and to be inconvenient to implement online or for continuing problems. In order to gain the advantages of bootstrapping in the case of policy gradient methods, we use actor–critic methods with a bootstrapping critic.

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: Let us assume that we have prior knowledge about the world. In some of the states, we know the next probable state for each action. Let S' be the set of all states with prior knowledge and S be the set of all states. Let us define $N(s,a)$ as the next probable state after taking action a in state $s \in S'$. $T(s,a)$, where $s \in S'$, is the sample mean of the rewards obtained during the transition $< s, a, N(s,a) >$. $n(s,a)$, where $s \in S'$, is the number of times the transition $< s, a, N(s,a) >$ occurs. We make the following adjustments to Q-learning:
a) If the current state $s_t \in S - S'$, we make a regular Q-learning update.
b) If the current state $s_t \in S'$, and the chosen action is a_t, we make a regular Q-learning update to $Q(s_t, a_t)$. In addition to that, for all $a \in A_{s_t}$ that is not equal to a_t and $n(s_t, a) \neq 0$, we perform the following update:
$Q(s_t,a) \leftarrow Q(s_t,a) + \alpha(T(s_t,a) + \gamma.max_{a'}Q(N(s_t,a),a') - Q(s_t,a))$

After performing the update, we update $n(s_t, a_t)$ and $T(s_t, a_t)$ accordingly.

In this way, we perform many updates with just a single sample from the environment.

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: $score(a) = \sum_s b_t(s).Q_t(s,a)$

where $b_t(s)$ is the belief associated with state s, $score(a)$ is the expected return for action a.
The policy is obtained by picking the action a that maximises the expected return. This is done by assuming that we know the information about the states. But because of the uncertainty in the POMDP, there might have been a better action to pick in terms of the total return. Since the partial observability is not taken into account, the obtained policy is not optimal for the POMDP. For the policy to be optimal, we must know all the states of the POMDP with certainty.

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: Condition 1: Max Node Irrelevance
Let M_i be a Max node in a MAXQ graph H for MDP M. A set of state variables Y is irrelevant to node i if the state variables of M can be partitioned into two sets X and Y such that for any stationary abstract hierarchical policy π executed by the descendants of i, the following two properties hold:
a) The state transition probability distribution $P^\pi(s', N|s, a)$ at node i can be factored into the product of two distributions:

$P^\pi(x', y', N|x, y, a) = P^\pi(y'|y, a).P^\pi(x', N|x, a)$
where y and y' give values for the variables in Y , and x and x' give values for the variables in X.

b) for any pair of states $s_1 = (x, y_1)$ and $s_2 = (x, y_2)$ such that $\chi(s_1) = \chi(s_2) = x$, and any child action a, $V^\pi(a, s_1) = V^\pi(a, s_2)$ and $\tilde{R}_i(s_1) = \tilde{R}_i(s_2)$.

Condition 2: Leaf Irrelevance
A set of state variables Y is irrelevant for a primitive action a of a MAXQ graph if for all states s the expected value of the reward function,
$V(a,s) = \sum_{s'} P(s'|s,a)R(s'|s,a)$

does not depend on any of the values of the state variables in Y . In other words, for any pair of states s_1 and s_2 that differ only in their values for the variables in Y,
$\sum_{s'_1} P(s'_1|s_1,a)R(s'_1|s_1,a) = \sum_{s'_2} P(s'_2|s_2,a)R(s'_2|s_2,a)$

Condition 3: Result Distribution Irrelevance
A set of state variables Y_j is irrelevant for the result distribution of action j if, for all abstract policies π executed by node j and its descendants in the MAXQ hierarchy, the following holds: for all pairs of states s_1 and s_2 that differ only in their values for the state variables in Y_j,
$P^{\pi}(s', N|s_1, j) = P^{\pi}(s', N|s_2, j)$

for all s' and N.
Condition 4: Termination
Let M_i be a task in a MAXQ graph such that for all states s where the goal predicate $G_i(s)$ is true, the pseudo-reward function $\tilde{R}_i(s) = 0$. Suppose there is a child task a and state s such that for all hierarchical policies π,
$\forall s',\ P_i^{\pi}(s', N|s, a) > 0 \Rightarrow G_i(s')$.

Then for any policy executed at node i, the completion cost $C(i,s,a)$ is zero and does not need to be explicitly represented.
Condition 5: Shielding
Let M_i be a task in a MAXQ graph and s be a state such that for all paths from the root of the graph down to node M_i there exists a subtask j (possibly equal to i) whose termination predicate $T_j(s)$ is true, then the Q nodes of M_i do not need to represent C values for state s.

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: