CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55
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. (2 marks) [Bandit Question] Consider a N-armed slot machine task, where the rewards for each arm a_i are usually generated by a stationary distribution with mean $Q^*(a_i)$. The machine is under repair when a arm is pulled, a small fraction, ϵ, of the times a random arm is activated. What is the expected payoff for pulling arm a_i in this faulty machine?

Solution:

We have N arms : $a_1, a_2, ..., a_N$. The mean of these arms are : $Q^*(a_1), Q^*(a_2), ..., Q^*(a_N)$
When arm a_i is pulled the following events can happen :
Case 1 : a_i gets activated, Probability = $(1-\epsilon) + \epsilon/N$
Case 2 : a_1 gets activated, Probability = ϵ/N
Case 3 : a_2 gets activated, Probability = ϵ/N
...
...
Case N : a_N gets activated, Probability = ϵ/N
Hence, the expected payoff for pulling arm a_i =

$$\left\{ \frac{\epsilon}{N} \sum_{j=1}^{N} Q^*(a_j) \right\} + (1-\epsilon)Q^*(a_i)$$

2. (4 marks) [Delayed reward] Consider the task of controlling a system when the control actions are delayed. The control agent takes an action on observing the state at time t. The action is applied to the system at time $t + \tau$. The agent receives a reward at each time step.

(a) (2 marks)What is an appropriate notion of return for this task?

Solution: Assumption : State remains the same in the absence of an action.
Trajectory : S_t, R_{t+1}, S_t, R_{t+2}, S_t, ..., $R_{t+\tau}$, S_t, a, $R_{t+\tau+1}$, $S_{t+\tau+1}$,...
Return : $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^\tau R_{t+\tau+1} + ...$

(b) (2 marks) Give the TD(0) backup equation for estimating the value function of a given policy.

Solution: TD target : $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^\tau R_{t+\tau+1} + \gamma^{\tau+1} V(S_{t+\tau+1})$
TD(0) backup equation :
$V(S_t) = V(S_t) + \alpha\{R_{t+1} + \gamma R_{t+2} + ... + \gamma^\tau R_{t+\tau+1} + \gamma^{\tau+1} V(S_{t+\tau+1}) - V(S_t)\}$

3. (5 marks) [Reward Shaping] Consider two finite MDPs M_1, M_2 having the same state set, S, the same action set, A, and respective optimal action-value functions Q_1^*, Q_2^*. (For simplicity, assume all actions are possible in all states.) Suppose that the following is true for an arbitrary function $f : S \to R$:

$$Q_2^*(s, a) = Q_1^*(s, a) - f(s)$$

for all $s \in S$ and $a \in A$.

(a) (2 marks) Show mathematically that M_1 and M_2 has same optimal policies.

Solution: We know that the optimal policy π^* for an MDP is the policy that maximizes the expected cumulative reward, which can be expressed in terms of the optimal action-value function Q^*. That is, for a given state s, the optimal policy chooses the action a that maximizes the value of $Q^*(s, a)$.
Let π_1^* and π_2^* be the optimal policies for M_1 and M_2, respectively. We want to show that $\pi_1^* = \pi_2^*$.
Since $Q_2^*(s, a) = Q_1^*(s, a) - f(s)$ for all $s \in S$ and $a \in A$, we can rewrite the value of the optimal policy for M_2 in terms of the optimal policy for M_1:

$$Q_2^*(s, a) = Q_1^*(s, a) - f(s)$$
$$= \max_\pi E_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a] - f(s)$$
$$= \max_\pi [E_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a] - f(s)]$$

where γ is the discount factor.

Since f is a function of s only, it does not depend on the action a. Therefore, the difference between the expected cumulative reward for M_1 and M_2 depends only on the value of $f(s)$ for each state s.

Let s be an arbitrary state in S. Then we have:

$$\pi_2^*(s) = \arg\max_a Q_2^*(s,a)$$

$$= \arg\max_a [E_{\pi_1^*}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a] - f(s)]$$

$$= \arg\max_a E_{\pi_1^*}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a]$$

$$= \arg\max_a Q_1^*(s,a)$$

$$= \pi_1^*(s)$$

where the last equality follows from the fact that Q_1^* is the optimal action-value function for M_1.

Therefore, we have shown that $\pi_2^*(s) = \pi_1^*(s)$ for all $s \in S$, which implies that M_1 and M_2 have the same optimal policies.

(b) (3 marks) Now assume that M_1 and M_2 has the same state transition probabilities but different reward functions. Let $R_1(s,a,s')$ and $R_2(s,a,s')$ give the expected immediate reward for the transition from s to s' under action a in M_1 and M_2, respectively. Given the optimal state-action value functions are related as given above, what is the relationship between the functions R_1 and R_2 ? That is, what is R_1 in terms of R_2 and f; OR R_2 in terms of R_1 and f.

Solution: We know that $Q_2^*(s,a) = Q_1^*(s,a) - f(s)$, for all $s \in S$ and $a \in A$.

Using the Bellman equation for the optimal action-value function, we can write:

$$Q_1^*(s,a) = R_1(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_1^*(s',a')$$

$$Q_2^*(s,a) = R_2(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_2^*(s',a')$$

We can substitute $Q_2^*(s,a)$ with $Q_1^*(s,a) - f(s)$ and simplify:

$$R_2(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A}(Q_1^*(s',a') - f(s'))$$
$$= R_1(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_1^*(s',a')$$

Rearranging and simplifying, we get:

$$R_2(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_1^*(s',a') - \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} f(s')$$
$$= R_1(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a' \in A} Q_1^*(s',a')$$

Since M_1 and M_2 have the same state transition probabilities, we can cancel out that term from both sides. Also, we can substitute $Q_2^*(s,a)$ with $Q_1^*(s,a) - f(s)$ and simplify further:

$$R_2(s,a,s') + \gamma \max_{a' \in A} \sum_{s' \in S} P(s'|s,a)(Q_1^*(s',a') - f(s'))$$
$$= R_1(s,a,s') + \gamma \max_{a' \in A} \sum_{s' \in S} P(s'|s,a) Q_1^*(s',a')$$

Again, since M_1 and M_2 have the same state transition probabilities, we can cancel out that term from both sides, and we get:

$$R_2(s,a,s') + \gamma \max_{a' \in A}(Q_1^*(s',a') - f(s'))$$
$$= R_1(s,a,s') + \gamma \max_{a' \in A} Q_1^*(s',a')$$

Finally, we can isolate $R_2(s,a,s')$ and get the relationship between R_1 and R_2:

$$R_2(s,a,s') = R_1(s,a,s') + \gamma \max_{a' \in A} Q_1^*(s',a') - \gamma \max_{a' \in A}(Q_1^*(s',a') - f(s'))$$

Simplifying further, we get:

$$R_2(s,a,s') = R_1(s,a,s') + \gamma f(s')$$

4. (10 marks) [Jack's Car Rental] Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$ 10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for

renting the day after they are returned. To help ensure that cars are available where they are needed, Jack can move them between the two locations overnight, at a cost of \$ 2 per car moved. We assume that the number of cars requested and returned at each location are Poisson random variables, meaning that the probability that the number n is $\frac{\lambda^n}{n!}e^{-\lambda}$, where λ is the expected number. Suppose λ is 3 and 4 for rental requests at the first and second locations and 3 and 2 for returns. To simplify the problem slightly, we assume that there can be no more than 20 cars at each location (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of five cars can be moved from one location to the other in one night.

(a) (4 marks) Formulate this as an MDP. What are the state and action sets? What is the reward function? Describe the transition probabilities (you can use a formula rather than a tabulation of them, but be as explicit as you can about the probabilities.) Give a definition of return and describe why it makes sense.

Solution: The main objective of the problem is to determine an optimal policy - the number of cars to be transferred from location 2 to location 1 (or vice versa) in order to maximise the expected profit.
Some mathematical notations to describe Jack's daily routine:
At the end of the day Jack counts N_1 cars at the first location(P_1) and N_2 cars at the second(P_2), so the **state** is $s = (N_1, N_2)$.
Next, Jack decides to take an **action** and move a cars from P_1 to P_2, which gets him to $(N_1^{''} = N_1 - a, N_2^{''} = N_2 + a)$ car arrangement at the start of the next day.
At the end of the next day at each location $X_i (i = 1, 2)$ cars were rented out and Y_i were returned. The **new state** at the end of next day is

$$s^{'} = (N_1^{'} = N_1^{''} - X_1 + Y_1, N_2^{'} = N_2^{''} - X_1 + Y_1)$$

The **reward(profit)** at the end of next day is

$$r = R_x \sum_{i=1,2} X_i + R_a |a|$$

where $R_x = 10$ and $R_a = -2$. According to the problem condition, each location does not fit more than $N_{max} = 20$ cars, so that $N_i, N_i^{'}, N_i^{''} \in [0, 20]$ and at most 5 cars can be moved so that $a \in [-5, 5]$. A negative value of a indicates a transfer of cars in the opposite direction, from P_1 to P_2.
Expected reward tensor $r(s, a)$ is a 3D tensor and can be evaluated once at the start. Since after the cars are transferred, the reward from each car park is statistically independent. For the **total reward** we can write

$$r((N_1, N_2), a) = R_x \sum_{X_1=0}^{N_1^{''}} X_1 p(X_1|\lambda_1, N_1^{''}) + R_x \sum_{X_2=0}^{N_2^{''}} X_2 p(X_2|\lambda_2, N_2^{''}) - R_a|a|$$

where

$$p(X|\lambda, N) = \begin{cases} p(X|\lambda) & X < N \\ \sum_{Z=N}^{\infty} p(Z|\lambda) & X = N \end{cases}$$

is the probability of X cars rented out from a location that had N cars at the start of a day. The probability of demand reaching X cars at a particular location with a mean demand λ follows the Poisson distribution.
Next, lets further simplify the **transition probability**. Like first two terms in the above equation for the expected reward, the transition probability depends on N_1, N_2, a only as

$$p((N_1^{'}, N_2^{'})|(N_1, N_2), a) = p((N_1^{'}, N_2{'})|(N_1^{''}, N_2^{''})) = p(N_1^{'}|N_1{''})p(N_1{'}|N_1{''}).$$

Therefore, the 5D tensor of transition probabilities $p(s^{'}|s, a)$ can be represented as a product of two 2D conditional probability tensors $p(N_i^{'}|N_i{''})$.

(b) (3 marks) One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs \$ 2, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of \$ 4 must be incurred to use a second parking lot (independent of how many cars are kept there). These sorts of nonlinearities and arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. Can you think of a way to incrementally change your MDP formulation above to account for these changes?

Solution: Let us see what happens if we add following non-linearities to the problem above :

- One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs \$2, as do all cars moved in the other direction.
- In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of \$4 must be incurred to use a second parking lot (independent of how many cars are kept there).

These conditions can be easily incorporated into the previous MDP formulation. We will have one more reward of -\$4 for the second parking lot if needed(Note that this reward is negative). Reward equation can be modified as follows -

$$r = \begin{cases} 10\sum_{i=1,2} X_i - 2(|a|-1) - \sum_{i=1,2} I_{N_i} \times 4 & a \in [1,5] \\ 10\sum_{i=1,2} X_i - 2(|a|) - \sum_{i=1,2} I_{N_i} \times 4 & a \in [-5,0] \end{cases}$$

I_{N_i} is indicator variable.

$$I_{N_i} = \begin{cases} 1 & \text{, when more than 10 cars are present in a location} \\ 0 & \text{, otherwise} \end{cases}$$

(c) (3 marks) Describe how the task of Jack's Car Rental could be reformulated in terms of *afterstates*. Why, in terms of this specific task, would such a reformulation be likely to speed convergence? *(Hint:- Refer page 136-137 in RL book 2nd edition. You can also refer to the video at https://www.youtube.com/watch?v=w3wGvwi336I)*

Solution: The current definition of state in Jack's Car Rental problem is a tuple, in which each entry indicates the number of cars in a location at the end of the day.
We can define the afterstates as a tuple, in which each entry is the number of cars in a location at the beginning of a day. That is to say, the states are defined after the action, i.e., car-moving during the night, are executed.
This reformulation is likely to speed convergence, as this reformulation reduces the dimension of states. Consider the case (i) at the end of a day, Jack has 4 cars in location A and 8 cars in location B, and he decides to transport 2 cars from B to A; case (ii) In the end of a day, Jack has 6 cars in location A and 6 cars in location B, and he decides to do nothing. The cases (i) and (ii) boils down to the same aftertstate, namely, (6, 6).

5. (8 marks) [Organ Playing] You receive the following letter:
Dear Friend, Some time ago, I bought this old house, but found it to be haunted by ghostly sardonic laughter. As a result it is hardly habitable. There is hope, however, for by actual testing I have found that this haunting is subject to certain laws, obscure but infallible, and that the laughter can be affected by my playing the organ or burning incense. In each minute, the laughter occurs or not, it shows no degree. What it will do during the ensuing minute depends, in the following exact way, on what has been happening during the preceding minute: Whenever there is laughter, it will continue in the succeeding minute unless I play the organ, in which case it will stop. But continuing to play the organ does not keep the house quiet. I notice, however, that whenever I burn incense when the house is quiet and do not play the organ it remains quiet for the next minute. At this minute of writing, the laughter is going on. Please tell me what

manipulations of incense and organ I should make to get that house quiet, and to keep it so.
Sincerely,
At Wits End

(a) (4 marks) Formulate this problem as an MDP (for the sake of uniformity, formulate it as a continuing discounted problem, with $\gamma = 0.9$. Let the reward be $+1$ on any transition into the silent state, and -1 on any transition into the laughing state.) Explicitly give the state set, action sets, state transition, and reward function.

Solution: State-space, $S = \{laughter\,(\,L\,),\ silent\,(\,S\,)\}$

Actions, $A = \{Playing\,Organ\,(O),\ Lighting\,Incense\,(I)\}$

Discount factor $\gamma = 0.9$.

Let us consider the following way of action-encodings. We will assume that if you burn incense, you won't play organ. And if you play organ, you don't burn incense. It doesn't matter as far as we get the optimal policy in the end. Also, encoding this way results in faster convergence of iterative methods.

The state transition diagram is given below.

$$\begin{tikzcd}
L \arrow[loop above, "{I,\,-1}"] \arrow[rr, "{O,\,+1}"] & & S \arrow[ll, bend right, "{O,\,-1}"'] \arrow[loop below, "{I,\,+1}"]
\end{tikzcd}$$

(b) (2 marks) Starting with simple policy of **always** burning incense, and not playing organ, perform a couple of policy iterations.

Solution: Let us assume an initial estimate of policy, $\pi_0 = \{L : I, S : I\}$. The symbols are defined in the previous part. Note that in vector notation, the order is: L, S (for states).

$$p_{\pi_0} = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \cdot\ r_{\pi_0} = \begin{matrix} -1 \\ +1 \end{matrix} \quad V_{\pi_0} = (I - \gamma p_{\pi_0})^{-1} r_{\pi_0} = \begin{matrix} -10 \\ 10 \end{matrix}$$

Now, $\pi_1(L) = \arg\max_a\{O : 1 + 0.9(10), I : -1 + 0.9(-10)\} = argmax_a\{O : 10, I : -10\} = O$
$\pi_1(S) = argmax_a\{O : -1 + 0.9(-10), I : 1 + 0.9(10)\} = argmax_a\{O : -10, I : 10\} = I$
Therefore, $\pi_1 = \{L : O,\ S : I\}$.

$$p_{\pi_1} = \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \cdot\ r_{\pi_1} = \begin{matrix} +1 \\ +1 \end{matrix} \quad V_{\pi_1} = (I - \gamma p_{\pi_1})^{-1} r_{\pi_1} = \begin{matrix} 10 \\ 10 \end{matrix}$$

Now, $\pi_2(L) = argmax_a\{O : 1 + 0.9(10), I : -1 + 0.9(10)\} = argmax_a\{O : 10, I : 8\} = O$
$\pi_2(S) = argmax_a\{O : -1 + 0.9(10), I : 1 + 0.9(10)\} = argmax_a\{O : 8, I : 10\} = I$
Therefore, $\pi_2 = \{L : O,\ S : I\}$.

$\pi_1 = \pi_2$. Thereforce, policy iteration converged to the optimal policy $\pi^* = \{L : O,\ S : I\}$.

And optimal value function $V^* = V_{\pi_1} = \{L : 10, S : 10\}$.

(c) (2 marks) Finally, what is your advice to “At Wits End”?

Solution: Advice to At Wits End is to follow the optimal policy as it ensures that the house remains silent. Since currently laughing sound is heard, At Wits End should play his organ and after that the house becomes silent. Now, he should switch and light an incense. He should keep doing this forever i.e he should keep burning incense and the house will be silent.

6. (4 marks) [Stochastic Gridworld] An ϵ-greedy version of a policy means that with probability 1-ϵ we follow the policy action and for the rest we uniformly pick an action. Design a stochastic gridworld where a deterministic policy will produce the same trajectories as a ϵ-greedy policy in a deterministic gridworld. In other words, for every trajectory under the same policy, the probability of seeing it in each of the worlds is the same. By the same policy I mean that in the stochastic gridworld, you have a deterministic policy and in the deterministic gridworld, you use the same policy, except for ϵ fraction of the actions, which you choose uniformly randomly.

(a) (2 marks) Give the complete specification of the world.

Solution: To design a stochastic gridworld where a deterministic policy will produce the same trajectories as an ϵ-greedy policy in a deterministic gridworld, we can create a world where the actions have equal probabilities of success and failure. Specifically, we can create a 2×2 gridworld with a start state at position $(1, 1)$ and a goal state at position $(2, 2)$. The actions available to the agent are 'up', 'down', 'left', and 'right', with equal probabilities of success and failure for each action. If the agent attempts to move in a certain direction, there is a 50% chance of successfully moving in that direction, and a 50% chance of moving in a random direction.

Here is the complete specification of the world:

- State space: { (1, 1), (1, 2), (2, 1), (2, 2) }
- Action space: { up, down, left, right }
- Initial state: (1, 1)
- Goal state: (2, 2)
- Dynamics: If the agent chooses an action a in state s:

  With probability 0.5, the agent transitions to the adjacent cell in the direction of a

  With probability 0.125, the agent transitions to each of the other adjacent cells

  With probability 0.25, the agent remains in the same cell
- Rewards: -1 for each time step until reaching the goal state, where the reward is 0

Now, we can use a deterministic policy for both the deterministic and stochastic gridworlds. Specifically, the policy is to always move to the right until reaching the goal state, and then stop. In the stochastic gridworld, we use an ϵ-greedy policy with $\epsilon = 0.5$, meaning that with probability 0.5, we follow the deterministic policy, and with probability 0.5, we choose a random action.

Since the actions have equal probabilities of success and failure, the probability of seeing any particular trajectory is the same for both the deterministic and stochastic gridworlds, regardless of whether the policy is deterministic or ϵ-greedy.

(b) (2 marks) Will SARSA on the two worlds converge to the same policy? Justify.

Solution: SARSA is an on-policy control algorithm that updates the Q-values based on the current policy. In both the deterministic and stochastic versions of the gridworld, SARSA will converge to the optimal policy as long as the learning rate is appropriately chosen and the exploration is sufficient. Therefore, SARSA will converge to the same policy in both versions of the gridworld. However, the convergence rate may be slower in the stochastic version due to the added randomness.

7. (5 marks) [Contextual Bandits] Consider the standard multi class classification task (Here, the goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs). Can we formulate this as contextual bandit problem (Multi armed Bandits with side information) instead of standard supervised learning setting? What are the pros/cons over the supervised learning method. Justify your answer. Also describe the complete Contextual Bandit formulation.

Solution: Yes, it is possible to formulate a multi-class classification task as a contextual bandit problem. In the standard supervised learning setting, the goal is to train a model to predict the correct class label given a set of input features. In the contextual bandit setting, the model must also decide which class label to predict based on the context.

To formalize the problem as a contextual bandit, we can define the context as the input features, and the set of possible actions as the set of class labels. At each iteration, the model observes the context and selects an action (i.e., a class label) to predict. The reward is then based on whether the predicted label matches the true label. The goal is to maximize the cumulative reward over time, which corresponds to maximizing the accuracy of the model.

The pros of using a contextual bandit approach over standard supervised learning include the following:

1. **Flexibility:** The contextual bandit approach allows for adaptive learning, where the model can update its predictions based on new data. This is particularly useful in settings where the distribution of the data changes over time, such as in online advertising.

2. **Efficiency:** The contextual bandit approach only needs to learn the optimal action in each context, rather than the entire decision boundary. This can lead to faster training times and lower computational requirements.

3. **Personalization:** The contextual bandit approach can be used to personalize the predictions for each user, based on their context and previous interactions.

The cons of using a contextual bandit approach include:

1. **Exploration-exploitation tradeoff:** The model needs to balance exploration (trying out new actions to learn about the environment) with exploitation (taking the currently best-known action). This can be challenging in practice, as choosing to explore may result in a lower reward in the short term.
2. **Complexity:** The contextual bandit approach can be more complex than standard supervised learning, as it requires a more intricate formulation of the problem and specialized algorithms.

In summary, the contextual bandit approach can be a useful alternative to standard supervised learning in certain settings, particularly those involving adaptive learning and personalization. However, it requires careful consideration of the exploration-exploitation tradeoff and specialized algorithms.

The complete contextual bandit formulation can be summarized as follows:

- **Context:** The set of input features that describe the current state of the environment.
- **Actions:** The set of possible actions that the model can take, based on the context.
- **Reward:** The feedback signal that the model receives after taking an action, which is typically based on how well the predicted label matches the true label.
- **Policy:** The decision-making function that maps the context to an action, based on the current estimate of the reward function.
- **Exploration:** The process of trying out new actions to learn about the environment and balance the exploration-exploitation tradeoff.
- **Exploitation:** The process of taking the currently best-known action, based on the current estimate of the reward function.

8. (5 marks) [TD, MC, PG] 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: If the system that we are trying to learn about is not perfectly Markov, that is, the Markov property does not hold exactly, then the suitability of different solution approaches will depend on the nature and extent of the deviation from the Markov property. In general, all three approaches, namely, Temporal Difference learning, Monte Carlo methods, and Policy Gradient algorithms, can be used in such cases, but each approach may have its own strengths and weaknesses.

- Temporal Difference (TD) learning can still be applied in cases where the Markov property holds approximately, but not perfectly. TD learning can capture some of the dependencies between states and the history of previous states and actions, which may be useful in such cases. However, if the deviation from the Markov property is significant, then TD learning may not be suitable.
- Monte Carlo methods can also be used in cases where the Markov property holds approximately. Monte Carlo methods do not make any assumptions about the Markov property and can learn from experience directly. However, Monte Carlo methods require complete episodes to be generated, which can be computationally expensive.
- Policy Gradient algorithms can also be used in cases where the Markov property holds approximately. Policy Gradient methods are model-free and can learn directly from experience, without making any assumptions about the Markov property. However, the efficiency of Policy Gradient algorithms may depend on the structure of the policy and the amount of variance in the policy's gradients.

In general, the suitability of these approaches will depend on the specific nature of the deviation from the Markov property. If the deviation is small, then TD learning may be the most suitable approach. If the deviation is significant, then Monte Carlo methods or Policy Gradient algorithms may be more appropriate. However, the choice of approach will depend on the specific problem and the available computational resources. It is also important to note that these approaches make certain assumptions about the underlying environment and may not always be applicable. Therefore, it is crucial to carefully assess the suitability of each approach before applying it to a given problem.

9. (5 marks) [PG] 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: The method described in the question can be considered a form of imitation learning, where the policy is learned by imitating an expert or a pre-defined behavior. It is **not a policy gradient method**, as it does not directly optimize the policy by maximizing a reward function.

In this method, the policy is represented as a labeling on the states, and a classifier is trained to predict the action associated with each state. The training data for the classifier consists of a few samples of state-action pairs, either provided by an expert or generated by an exploration strategy.

Once the classifier is trained, it can be used to generate a policy by predicting the action for each state. This policy can then be evaluated using a suitable reward function, and refined using reinforcement learning techniques if necessary.

To generate appropriate targets for the classifier, one approach is to use the expert's actions as the target labels. In this case, the training data consists of pairs of states and expert actions, and the classifier is trained to predict the expert action for each state.

Another approach is to use a pre-defined behavior as the target policy, such as a hand-crafted rule-based policy or a previously learned policy. In this case, the training data consists of pairs of states and actions generated by the target policy, and the classifier is trained to predict the target action for each state.

In both cases, the quality of the learned policy depends on the quality of the training data and the ability of the classifier to generalize to unseen states. The method can be improved by incorporating exploration strategies to collect more diverse training data, and by using more powerful classifiers that can handle complex state-action mappings.