CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55
Name: K.S.MOHAN KUMAR **Roll number:** ME19B118

- 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: Here, we use the E[X]=E[E[X/Y]]. Y represents the event that, whether the machine is under repair or not.X represent the payoff.

$$\mathbb{E}[\mathbb{X}] = (1-\epsilon) * (Q^*(a_i)) + \epsilon * (\frac{\Sigma_{j=1}^{n}(Q^*(a_j))}{n}) \tag{1}$$

When it is assumed that, during random selection the pulled arm does not contribute to the expected reward,

$$\mathbb{E}[\mathbb{X}] = (1-\epsilon-\frac{\epsilon}{n}) * (Q^*(a_i)) + \epsilon * (\frac{\Sigma_{j=1}^{n}(Q^*(a_j))}{n}) \tag{2}$$

Remarks: Other possible cases:
After an arm is selected, it returns to its original position and during random another random arm gets selected.

$$\mathbf{E}[\mathbf{X_i}] = (1-\epsilon-\frac{\epsilon}{n}) * (Q^*(a_i)) + \epsilon * (\frac{\Sigma_{j=1}^{n}(E(X_j))}{n})$$

Solving the n linear equations will get us the desired value

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: Here, assuming the reward distribution is of the form $\mathcal{R} : \mathcal{S}, \mathcal{A} \Longrightarrow \mathbb{R}$

   Since the rewards are obtained at each time step,

   $$\mathbb{G}_t = \mathbf{R_{t+\tau+1}} + \gamma * \mathbf{R}_{t+\tau+2} + \gamma^2 * \mathbf{R}_{t+\tau+3} + \gamma^3 * \mathbf{R}_{t+\tau+4}... \tag{3}$$

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

   Solution:

   $$V(S_t) \longleftarrow V(S_t) + \alpha \cdot (\mathbf{R_{t+\tau+1}} + \gamma \cdot V(S_{t+1}) - V(S_t))$$

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: Let the optimal policy for M1 be π_1^* and for M2 be π_2^*. Since, it is given that Q_1^* and Q_2^* are optimal action-value functions, from the bellman equations,
   we can write

   $$\pi_1^*(s) = \arg\min_A(Q_1^*(s, a)) \tag{4}$$

   $$\pi_2^*(s) = \arg\min_A(Q_2^*(s, a)) \tag{5}$$

$$\pi_2^*(s) = \arg\min_A(Q_1^*(s,a) - f(s)) \tag{6}$$

Since, f(s) is independent of actions at a particular state in (3) we are just subtracting a constant value for all action-value functions in a state. Hence, the argmin over all the actions for $\pi_1^*(s)$ and $\pi_2^*(s)$ will be the same.
Since f(s) is a constant it can be removed out of the argmin.

$$\pi_2^*(s) = \arg\min_A(Q_1^*(s,a)) \tag{7}$$

So, both have 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: From the Bellmann Equation of optimality,

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} \mathbf{P}(s', r|s, a)(r + \gamma \max_{a' \in \mathcal{A}(s')} Q^*(s', a'))$$

Since R1 and R2 represent the expected rewards,

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} \mathbf{P}(s'|s, a)(R(s,a,s') + \gamma \max_{a' \in \mathcal{A}(s')} Q^*(s', a'))$$

For M1,

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

For M2,

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

Subtracting the equations and substituting the relationship between Q1 and Q2,

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

since max term does not have any effect on f(s') we can remove it.
Since f(s) is independent of $\sum_{s' \in \mathcal{S}} \mathbf{P}$,

$$\sum_{s' \in \mathcal{S}} \mathbf{P}(f(s)) = 1$$

Using the above relation,

$$0 = \sum_{s' \in \mathcal{S}} \mathbf{P}(s'|s,a)((R_1(s,a,s') - R_2(s,a,s') + \gamma f(s') - f(s)))$$

Thus,

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

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: This problem can be formulated as a finite horizon Markov Decision Process (MDP) with a daily time step.
State Space: The state of the MDP is defined by the number of cars at each location, which can range from 0 to 20 for each location. $\mathbf{S_t}$=(C_1, C_2), where C_1 represents the number of cars in 1 at the end of the day.

Action Space: The action space is the set of all possible moves of cars between the two locations. The agent can decide to move up to 5 cars from one location

to the other, or not move any cars.A=[-5,5] where negative values indicate that a car is moved from b to a and positive values indicate vice versa.

Reward Function: The reward function is defined as follows: when a car is rented out, Jack receives a reward of \$10 .When he moves a car between the two locations, he incurs a cost of \$2 per car moved. The goal of the MDP is to maximize the expected cumulative reward over a finite horizon.

Let X1 and X2 be the number of rental requests at location 1 and 2, respectively, and let Y1 and Y2 be the number of returns at location 1 and 2, respectively.

$$R_t = -2 \cdot |a| + 10 \cdot (min(X1, C_1 - a) + min(X2, C_2 + a)$$

TransitionProbabilities : The transition probabilities are the probabilities of moving from one state to another given an action. The transition probability depends on the number of cars requested and returned at each location, as well as the action taken by Jack. Specifically, if Jack decides to move n cars from Location 1 to Location 2, then the number of cars at each location at the end of the day is determined by the following formula: Let X1 and X2 be the number of rental requests at location 1 and 2, respectively, and let Y1 and Y2 be the number of returns at location 1 and 2, respectively. Let $(C_1^{'}, C_2^{'})$ be the next state transition after taking an action a.

$$\begin{aligned}
P(C_1^{'}, C_2^{'}|C_1, C_2, a) &= P(C_1^{'}|C_1, a) \cdot P(C_2^{'}|C_1, a) \\
C_1^{'} &= min(C_1 + Y1 - a - X1, 20) \\
C_2^{'} &= min(C_2 + Y2 + a - X2, 20) \\
P(C_1^{'}|C_1, a) &= \Sigma_{x=0}^{C_1 - a - C_1^{'}} \frac{e^{-3}3^x}{x!} \cdot \frac{e^{-3}3^{C_1^{'} + x + a - C_1}}{(C_1^{'} + x + a - C_1)!} \\
P(C_2^{'}|C_2, a) &= \Sigma_{x=0}^{C_2 + a - C_2^{'}} \frac{e^{-4}4^x}{x!} \cdot \frac{e^{-2}2^{C_2^{'} + x - a - C_2}}{(C_1^{'} + x - a - C_2)!} \\
&Here,
\end{aligned}$$

Return Definition: The return is the cumulative discounted reward obtained by the agent over a finite horizon. Specifically, the return at time t is defined as:

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} R_{i+1}$$

We should ensure that both locations dosen't run out of cars and maximising the expected return.

where T is the final time step, γ is a discount factor, and R_{i+1} is the reward obtained by the agent at time $i+1$. The goal of the agent is to choose actions that maximize the expected cumulative return. Discounted return induces a sense of future rewards with giving importance to current rewards helps keep the business in operation for a long time.

(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: These changes can be incorporated into the reward function.
Moving Cost= $\begin{cases} 2\cdot \text{(a-1)} & \text{if a}> 0 \\ 2\cdot|a| & \text{if a}\leq 0 \end{cases}$
Moving Cost=$2\cdot \min(|a|, |a-1|)$
Parking Cost= 4*(max(min($C_a - a - 10 - |C_a - a - 10|$,1),0)+max(min($C_b + a - 10 - |C_b + a - 10|$,1),0))
We dont have to consider the no of cars being more than 20 here, as we are just checking whether the no of cars is greater than 10.

So, $\text{R}_t = -2 \cdot \min(|a|, |a-1|) + 10 \cdot (min(X1, C_a - a) + min(X2, C_b + a) + 4 * (max(min(C_a - a - 10 - |C_a - a - 10|, 1), 0) + max(min(C_b + a - 10 - |C_b + a - 10|, 1), 0))$

(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: Here, we can model the afterstates as the number of cars present at the start of the day, where the shifting of cars take place in the night. More Formally, $S^{'}$=$\{\text{C}_a - a, C_b + a\} = \{N1, N2\}$
$Nowallthepairs(\text{C}_a, a)$ corresponding to a particular N1 are considered as the same state. Similar, argument holds for N2.

Thus the dimensions in the problem decreases and this reformulation increases the speed of convergence as the state space decreases.

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={Laughter,Quiet}
Playing Organ- PO, Burning Incense-BI
Actions={PO,BI,PO$\wedge BI, \neg PO \wedge \neg BI$}

$$Rewards = \begin{cases} +1 & \text{for any transition into the silent state} \\ -1 & \text{for any transition into the laughing state} \end{cases}$$

Solution:

Transition Probability Matrix				
Current State	Next State	Action	P(s'—s,a)	Reward
L	L	PO	0	-1
L	L	BI	1	-1
L	L	PO$\wedge BI$	0	-1
L	L	$\neg PO \wedge \neg BI$	1	-1
L	Q	PO	1	1
L	Q	BI	0	1
L	Q	PO$\wedge BI$	1	1
L	Q	$\neg PO \wedge \neg BI$	0	1
Q	L	PO	1	-1
Q	L	BI	0	-1
Q	L	PO$\wedge BI$	0	-1
Q	L	$\neg PO \wedge \neg BI$	1	-1
Q	Q	PO	0	1
Q	Q	BI	1	1
Q	Q	PO$\wedge BI$	1	1
Q	Q	$\neg PO \wedge \neg BI$	0	1

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

Solution:

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

Performing Policy Iteratation:
Iteration1
Policy evaluation:
v(L)=v(Q)=0; $\pi(BI|L) = \pi(BI|Q) = 1$
Let $\theta = 0.5, \Delta = 0$
v(L) =(-1+0.9*0)=-1 v(Q)=(1+0.9*0)=1 $\Delta = 1$
v(L) =(-1+0.9*-1)=-1.9 v(Q)=(1+0.9*0)=1.9 $\Delta = 0.9$
v(L) =(-1+0.9*-1.9)=-2.71 v(Q)=(1+0.9*0)=1=2.71 $\Delta = 0.81$
v(L) =(-1+0.9*0)=-3.439 v(Q)=(1+0.9*0)=3.439 $\Delta = 0.729$
v(L) =(-1+0.9*0)=-4.0951 v(Q)=(1+0.9*0)=4.0951 $\Delta = 0.6561$
v(L) =(-1+0.9*0)=-4.68559 v(Q)=(1+0.9*0)=4.68559 $\Delta = 0.59$
v(L) =(-1+0.9*0)=-5.21 v(Q)=(1+0.9*0)=5.21 $\Delta = 0.524$

v(L) =(-1+0.9*0)=-5.69 v(Q)=(1+0.9*0)=5.69 $\Delta = 0.48$

Policy Improvement:
$\pi(L)$=PO (Ties broken with PO $\wedge BI$)
$\pi(Q)$=BI

Iteration2: Policy Evaluation: v(L)=(1+0.9*(5.68))=6.121 v(Q)=1+0.9*5.68=6.121
$\Delta = 0.431$
Policy Improvement:
$\pi(L)$=PO
$\pi(Q)$=BI

Since the policy is stable, we break.

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

Solution: If there is laughter, play the organ; if room is quite, do not play the organ and burn incense.

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: Assume the number of actions possible in the grid world is n.Let s^d be the state moved to by the agent by taking a^d where a^d represents the action take according to the deterministic policy.
In the deterministic world, using the epsilon-greedy version of a policy,

$$\mathcal{P}(a) = (1-\epsilon)\cdot(\pi(a|s)) + \frac{\epsilon}{n}$$

We have to make the state transition probabilities equal to that obtained from the deterministic gridworld to get the same trajectories.**We have to convert**

the stochasticity in choosing actions in the gridworld to stochsticity in the environment. We can model this by

$$\mathcal{P}(s^{'}|s,a) = (1-\epsilon)\cdot(\pi(a|s)) + \frac{\epsilon}{n}$$

$$\pi(a|s) = \begin{cases} 1, & \text{if } s' = s^d \\ 0, & s'! = s^d \end{cases}$$

$$\mathcal{P}(s^{'}|s,a) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{n}, & \text{if } s' = s^d \\ \frac{\epsilon}{n}, & s'! = s^d \end{cases}$$

Here s^d represents the state reached when taking an action a in the deterministic world.

A deterministic policy is a function of the form $\pi_d : S \to A$, that is, a function from the set of states of the environment, S, to the set of actions, A. The subscript d only indicates that this is a deterministic policy.

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

Solution: The update rule for SARSA is

$$Q(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha\cdot(r + \gamma Q(s',a')) \tag{8}$$

No SARSA on the two worlds is not guranteed converge to the same policy.During the update step,the stochasticity for stochastic gridworld is loaded into the next state, whereas in the deterministic world, the stochasticity is loaded with next actions. Hence the updates are affected differently.

An alternative view:
From Part 1, for every trajectory in stochastic gridworld is possible in the deterministic gridworld with same probaility. But since sampling the same trajectories happen at different times, the update rule gets affected which can alter the final policy. This can be prominently seen when there are obstacles in the grid world/ multiple termination states.

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, the standard multi-class classification task can be formulated as a contextual bandit problem, also known as the multi-armed bandit problem with side information.

In the contextual bandit setting, the learner is presented with a set of arms, each corresponding to a different action or class label. For each action, the learner also receives a set of features or context information that describes the state of the environment. The learner's goal is to choose the best action based on the context information and receive a reward based on the correctness of the action.

In the case of multi-class classification, each class corresponds to an arm and the context information is the feature vector associated with the data point. The learner must choose the class label that maximizes the probability of correct classification and receive a reward based on the correctness of the prediction.

Pros and Cons:
The advantage of the contextual bandit approach over supervised learning is that it can handle situations where the cost of labeling data is high or the training data is scarce. In the contextual bandit setting, the learner can actively choose which data points to label and update its policy in real-time based on the rewards received. This can result in faster convergence and better generalization to new data.Online learning is possible as policies get updated when new context vectors are witnessed.

We have the usual problem of explore-exploit dilemma.The model needs to observe many context vectors to get a comparable accuracy to that of supervised learning.When there are many contexts the dimensionality of the reward function increases. These are solved using CNNs in case of standard supervised learning.

The complete contextual bandit formulation involves the following components:

Action space: The set of possible actions or class labels.

Context space: The set of feature vectors associated with each data point.

Policy: A function that maps each context vector to a probability distribution over the action space.

Reward function: A function that maps each (context, action) pair to a reward value based on the correctness of the action.

Exploration strategy: A mechanism for exploring the action space to learn the best policy.

The goal of the learner is to learn a policy that maximizes the expected cumulative reward over time. This can be achieved through various algorithms such as epsilon-greedy, UCB, and Thompson sampling, which balance the exploration of new actions with the exploitation of the best action according to the current policy.

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 being learned is not perfectly Markov, i.e., the current state does not contain all the information necessary to determine the future states and rewards, then different solution approaches may be more or less suitable depending on the specifics of the system.

THe TD algorithm estimates the state value function under a policy. This implies that we need a Markov model to estimate the transition probabilitites.

The one step TD target that is sampled in basic TD learning is simply the inner part of this:

$Gt = R_{t+1} + V(S_{t+1})$

which when sampled is equal to v(s) in expectation *, when St=s . That is, when you measure a single instance of the TD target and use it to update a value function, you implicitly assume that the values or r(t+1) and s(t+1) that you observed occur with probabilities determined by $\pi(a|s)$ and $p(r, s|s, a)$ as shown by the Bellman equation.

Monte Carlo methods,are model-free reinforcement learning algorithms that do not assume a Markovian structure, can be used in cases where the system has a larger state space and the dynamics of the system are more difficult to model. Monte Carlo methods can handle non-linear dynamics and are robust to errors in the transition probabilities. However, Monte Carlo methods require complete trajectories of the system, which may be difficult or costly to obtain.

Policy Gradient (PG) algorithms are model-free reinforcement learning algorithms that learn the policy directly by optimizing a performance objective. PG algorithms are well-suited for non-Markovian systems since they can handle partial observability and non-linear dynamics. PG algorithms can also handle continuous action spaces, making them more suitable for tasks such as control.

However, PG algorithms assume that the policy is differentiable with respect to its parameters, and that the optimization problem can be solved efficiently. If the policy space is large or the optimization problem is intractable, PG algorithms may not perform well.

In summary, the suitability of TD learning, Monte Carlo methods, and Policy Gradient algorithms depends on the specifics of the system being learned. TD learning methods are more suitable for smaller state spaces and Markovian systems, while Policy Gradient algorithms are more suitable for larger state spaces and non-Markovian systems.

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: No, the method of turning the reinforcement learning problem into a classification problem cannot be considered as a policy gradient method. Policy gradient methods are a class of reinforcement learning algorithms that learn the policy directly by optimizing a performance objective. In contrast, the method of turning the reinforcement learning problem into a classification problem involves learning the policy indirectly by training a classifier to predict the policy labels from the states.It does not optimize a parameterized policy using gradient descent. Instead, it treats the policy as a labeling on the states and uses a classifier to learn the labels. The policy evaluation stage may still involve using gradient descent to optimize the policy, but the initial training phase does not.

A complete method for generating appropriate targets for the classifier involves the following steps:

Collect a set of state-action pairs by following the current policy in the environment. Use these state-action pairs to label the corresponding states as either "good" or "bad" based on the reward obtained from the environment. Train a classification model on this labeled data to predict the policy labels for each state. Use the trained classifier to generate a new policy by selecting the action that is most likely to lead to a "good" state. The success of this method depends on the quality and size of the labeled data used to train the classification model. Therefore, it is important to

collect a diverse set of state-action pairs and to label them accurately to obtain an effective policy.