CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:55

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. (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:

The given bandit task is stochastic due to **two** reasons:

1. The reward on the activation of any arm is stochastic. This is the case with any bandit task.
2. The arm which is activated might be different from the one that is pulled, making this stochastic. This is unique to this particular bandit task.

Taking both these into account, the expected payoff for pulling arm a_i, which we denote as P_{a_i}, in this case can be written as:

$$P_{a_i} = \sum_{j=1}^{N} \mathbb{E}(r_{a_j}) p_{a_j}$$

where $\mathbb{E}(r_{a_j})$ is the expected reward on *activation* (not pulling) of arm a_j and hence is equal to $Q^*(a_j)$ and p_{a_j} is the probability of activation of arm a_j given that arm a_i is pulled.

As per the given conditions,

$$p_{a_j} = \begin{cases} 1 - (\frac{N-1}{N})\epsilon, & \text{if } j = i \\ \frac{\epsilon}{N}, & \text{otherwise} \end{cases}$$

Therefore,

$$\boxed{P_{a_i} = Q^*(a_i)(1 - (\frac{N-1}{N})\epsilon) + \sum_{j \neq i} Q^*(a_j)\frac{\epsilon}{N}}$$

where i and j go from 1 to N.

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:
   Return is a metric defined for the purpose of identifying how good a course of action is in the long term. Hence, ideally, return following an action should indicate how good the action is in the long term.
   In the context of delayed rewards, it is therefore necessary to identify **only the rewards which result from the action** while computing the return. In this case, the rewards for an action at time t follow only at time $t + \tau$. Hence, an appropriate definition of return at time t, G_t, in this case would be:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+\tau+k+1}$$

   where γ is the discount factor and R_j is the reward at time j.

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

   Solution:
   Let us denote the value function for a given policy π as v_π. Let $v_\pi(s_t)$ denote the value function for the state s_t at time t and $v_\pi(s_{t+1})$ denote the value function for the state s_{t+1} at time $t + 1$ where s_{t+1} is obtained by following policy π. Then we have:

$$
\begin{aligned}
v_\pi(s_t) &= \mathbb{E}_\pi(G_t|s_t) \\
&= \mathbb{E}_\pi(\sum_{k=0}^{\infty} \gamma^k R_{t+\tau+k+1}|s_t) \\
&= \mathbb{E}_\pi(\sum_{k=0}^{\infty} \gamma^k R_{t+\tau+k+1}|s_t) \\
&= \mathbb{E}_\pi(R_{t+\tau+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+\tau+k+1}|s_t) \\
&= \mathbb{E}_\pi(R_{t+\tau+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{(t+1)+\tau+k+1}|s_t)
\end{aligned}
$$

But clearly,

$$\mathbb{E}_\pi(\sum_{k=0}^{\infty} \gamma^k R_{(t+1)+\tau+k+1}|s_t) = v_\pi(s_{t+1})$$

Therefore,

$$v_\pi(s_t) = \mathbb{E}(R_{t+\tau+1} + \gamma v_\pi(s_{t+1}))$$

In TD(0) backup, we sample and bootstrap based on this equation. Therefore the TD(0) update rule becomes:

$$\boxed{v_\pi(s_t) \leftarrow v_\pi(s_t) + \alpha(R_{t+\tau+1} + \gamma v_\pi(s_{t+1}) - v_\pi(s_t))}$$

where α is the learning rate.

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:
By definition, a policy which is greedy on optimal action value function is an optimal policy. Let us denote an optimal policy for M_1 as π_1^* and an optimal policy for M_2 as π_2^*. Then we have:

$$\pi_1^*(a|s) = \arg\max_{a'} Q_1^*(s, a')$$
$$\pi_2^*(a|s) = \arg\max_{a'} Q_2^*(s, a')$$

But given

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

Therefore,

$$\begin{aligned}\pi_2^*(a|s) &= \arg\max_{a'}(Q_1^*(s, a') - f(s)) \\ &= \arg\max_{a'}(Q_1^*(s, a') \\ &= \pi_1^*(a|s)\end{aligned}$$

since $f(s)$ is independent of a. Therefore 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:
Let us denote the common state transition probabilities for M_1 and M_2 as $p(s'|s, a)$. We have shown in the above part that both M_1 and M_2 have the same optimal policies. Let us denote one such optimal policy as π^*. Then the Bellman optimality equations for M_1 takes the form:

$$Q_1^*(s, a) = \sum_{s'} p(s'|s, a)[R_1(s, a, s') + \gamma \max_{a'} Q_1^*(s', a')] \tag{1}$$

But we have,

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

Therefore (1) takes the form:

$$Q_2^*(s,a) + f(s) = \sum_{s'} p(s'|s,a)[R_1(s,a,s') + \gamma(\max_{a'} Q_2^*(s',a') + f(s'))] \quad (2)$$

Note that we can take $f(s')$ outside the max function since it is independent on a'. Also note that:

$$\sum_{s'} p(s'|s,a)f(s) = f(s) \quad (3)$$

since $f(s)$ is independent of s' and $\sum_{s'} p(s'|s,a) = 1$. Hence, we can rewrite (2) as:

$$Q_2^*(s,a) + \sum_{s'} p(s'|s,a)f(s) = \sum_{s'} p(s'|s,a)[R_1(s,a,s') + \gamma(\max_{a'} Q_2^*(s',a') + f(s'))]$$

which reduces to:

$$Q_2^*(s,a) = \sum_{s'} p(s'|s,a)[(R_1(s,a,s') + \gamma f(s') - f(s)) + \gamma \max_{a'} Q_2^*(s',a'))] \quad (4)$$

The Bellman optimality equation for M_2 is:

$$Q_2^*(s,a) = \sum_{s'} p(s'|s,a)[R_2(s,a,s') + \gamma \max_{a'} Q_2^*(s',a')] \quad (5)$$

Since (4) and (5) hold for all state action pairs (s,a), we must have:

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

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:

State Set: $s \in \{(n_1, n_2)\},$ where $0 \leq n_1 \leq 20,$ and $0 \leq n_2 \leq 20.$
Here, n_1 represents the number of cars currently at location 1 and n_2 represents the number of cars currently at location 2. Since these numbers must lie between 0 and 20, we have 441 total states.

Action Set: $a \in [-5, 5], a \in \mathbb{Z}.$
Here, a corresponds to the number of cars transferred from location 1 to location 2. We represent cars transferred from location 2 to location 1 by using negative values for a.

Reward Function: $R = -2|a| + 10(\min(rq_1, n_1) + \min(rq_2, n_2)).$
where rq_1 and rq_2 denote the number of cars requested at location 1 and location 2 respectively. Note that only upto the total number of cars in that location can be rented out. $|a|$ corresponds to the number of cars transported at the end of that day. Also, rq_1 and rq_2 are Poisson random variables with mean 3 and 4 respectively.

Transition probabilities: Consider the transition: $(n_1, n_2) \rightarrow (n_1', n_2').$
We are required to compute the quantity $p((n_1', n_2')|\ (n_1, n_2), a)$. From the multiplication theorem in probability:

$$p((n_1', n_2')|\ (n_1, n_2), a) = p(n_1'|\ n_1, a)p(n_2'|\ n_2, a)$$

From the definition of the problem, the number of cars in location 1 after one day (n_1'):

$$n_1' = n_1 - \min(rq_1, n_1) - a + rt_1$$

where the new term rt_1 denotes the number of cars returned that day (these are ready to use the next day and hence contribute to the count of next state). Similarly,

$$n_2' = n_2 - \min(rq_2, n_2) + a + rt_2$$

We can write:

$$rt_1 = (n_1' - n_1 + a + \min(rq_1, n_1))$$

and for location 2:

$$rt_2 = (n_2' - n_2 - a + \min(rq_2, n_2))$$

Note that n_1, n_2 and a are deterministic constants once a state and action are chosen, and n_1' and n_2' are deterministic once the transition probability to be computed is fixed. Also, we only have positive probabilities for actions which can be taken, i.e., number of cars should be sufficient for transfer to take place. For now, let us assume that n_1' and n_2' are less than 20. Taking into account the fact that requests and returns are Poisson, we have:

$$p(n_1'|\ n_1, a) = \sum_{rq_1=0}^{\infty} \frac{3^{rq_1}e^{-3}}{rq_1!} \frac{3^{(n_1'-n_1+a+\min(rq_1,n_1))}e^{-3}}{(n_1'-n_1+a+\min(rq_1,n_1))!}$$

$$p(n_2'|\ n_2, a) = \sum_{rq_2=0}^{\infty} \frac{4^{rq_2}e^{-4}}{rq_2!} \frac{2^{(n_2'-n_2-a+\min(rq_2,n_2))}e^{-2}}{(n_2'-n_2-a+\min(rq_2,n_2))!}$$

For the case of $n_1' = 20$ and $n_2' = 20$ we have:

$$p(20|\ n_1, a) = \sum_{i=0}^{\infty}\sum_{rq_1=0}^{\infty} \frac{3^{rq_1}e^{-3}}{rq_1!} \frac{3^{(n_1'-n_1+a+\min(rq_1,n_1))+i}e^{-3}}{((n_1'-n_1+a+\min(rq_1,n_1))+i)!}$$

$$p(20|\ n_2, a) = \sum_{i=0}^{\infty}\sum_{rq_2=0}^{\infty} \frac{4^{rq_2}e^{-4}}{rq_2!} \frac{2^{(n_2'-n_2-a+\min(rq_2,n_2))+i}e^{-2}}{((n_2'-n_2-a+\min(rq_2,n_2))+i)!}$$

Return: $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Since we do not have any fixed terminal state, it makes sense to use discounted return as opposed to undiscounted return. However, we would still like to keep the value of γ sufficiently so as to ensure higher long term rewards. This is necessary because, though the action of shifting cars can temporarily decrease rewards, it is crucial for obtaining higher rewards in the longer run by ensuring sufficient supply in both locations. By setting γ appropriately, once can thus maximize long term rewards.

(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:

The effect of Jack's employee shifting a car can be incorporated by modifying the reward function as follows:

$$R_{emp} = -2\min(|a|, |a-1|) + 10(\min(rq_1, n_1) + \min(rq_2, n_2))$$

This works because for all values of a greater than 1, Jack's employee can help shift one car thereby reducing the count by 1.

The number of cars that has to be kept overnight after accounting for moving is nothing but the number of cars in the next state, i.e., n'_1 and n'_2. Therefore the extra cost of parking lot can be incorporated as:

$$R_{lot} = R_{emp} - 4sgn(n'_1 - 10) - 4sgn(n'_2 - 10)$$

where

$$sgn(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

(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:

An afterstate refers to the set of states (may be different from the original state space) which the agent ends up in after accounting for the stochasticities not under the control of the agent. In this particular problem, stochasticity is introduced by the number of cars that are requested and returned at the two locations. Therefore, the after states here would be the car counts at locations 1 and 2 after adding and subtracting the number of cars returned and requested respectively. More precisely, the set of afterstates:

$$S' = \{(m_1, m_2)\}$$

$$\text{where } m_1 = n_1 - rq_1 + rt_1$$

$$\text{and } m_2 = n_2 - rq_2 + rt_2$$

In this task, from a given afterstate, the best afterstate on transition can be computed deterministically since now you only have to account for the agent's action, which is something under your control. This will likely lead to faster convergence since this is more straightforward as compared to state transitions which are stochastic due to the car requests and returns.

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 set: {Silent (S), Laughing (L)}

Action set: Although you have four total combinations of playing organ and burning incense, the following two actions are sufficient to describe the given problem:

{Play organ(O), Do not play organ but burn incense (I)}

State transitions and rewards: Note that state transitions and rewards are deterministic once you fix your current state and action. Hence they can be represented in the following tabular format:

Current state	Action	Next state	Reward
S	O	L	-1
S	I	S	+1
L	O	S	+1
L	I	L	-1

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

Solution:

Since we are dealing with very small state and action spaces, we can directly do policy evaluation by solving Bellman's equations instead of updating iteratively. Policy improvement is then done by being greedy with respect to the value functions.

Policy Iteration 1:

We start with the policy of always picking action I. We denote the value functions for state S and state L for this policy as $v_I(S)$ and $v_I(L)$ respectively.

Policy Evaluation:

$$\begin{aligned} v_I(S) &= 1 + \gamma v_I(S) \\ &= 1 + 0.9 v_I(S) \\ \implies v_I(S) &= 10 \\ v_I(L) &= -1 + \gamma v_I(L) \\ &= -1 + 0.9 v_I(L) \\ \implies v_I(L) &= -10 \end{aligned}$$

Policy Improvement:

We evaluate:

$$a = \arg\max_{a'} R(s, a', s') + \gamma v_I(s')$$

for each state s where s' is the state after transition.

For state S:

For action I:

$$R(s, a', s') + \gamma v_I(s') = 1 + 0.9 * 10 = 10$$

For action O:

$$R(s, a', s') + \gamma v_I(s') = -1 + 0.9 * (-10) = -10$$

So we choose action I in state S.

For state L:
For action I:

$$R(s, a', s') + \gamma v_I(s') = -1 + 0.9 * (-10) = -10$$

For action O:

$$R(s, a', s') + \gamma v_I(s') = 1 + 0.9 * (10) = 10$$

So we choose action O in state L.

Policy Iteration 2:

The new policy is to choose I in S and O in L. We denote this policy as π.
Policy Evaluation:

$$\begin{aligned} v_\pi(S) &= 1 + \gamma v_\pi(S) \\ &= 1 + 0.9 v_\pi(S) \\ \implies v_\pi(S) &= 10 \\ v_\pi(L) &= 1 + \gamma v_\pi(S) \\ &= 1 + 0.9 v_\pi(S) \\ \implies v_\pi(L) &= 10 \end{aligned}$$

Policy Improvement:
Clearly, 10 is the highest possible value for the value functions. Hence, the policy π is greedy w.r.t its own value function. **Therefore, π, which says to choose I in S and O in L, is optimal.**

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

Solution:
Clearly, we should follow the optimal policy π derived in the above section.
Hence, play the organ initially when you are at Laughter state. This transitions you into the Silent state. Keep burning the incense without playing the organ in the Silent state so that you remain in this state.

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:

The four available actions are still {Up, Left, Down, Right}. However, on choosing an action, the agent moves in the direction of prescribed action only with probability $1 - \frac{3\epsilon}{4}$ and has an equal probability of $\frac{\epsilon}{4}$ for moving in any one of the other three directions.

As an example, if the agent performs the action Up, it has a probability of $1 - \frac{3\epsilon}{4}$ for moving upwards and a probability of $\frac{\epsilon}{4}$ each for moving leftwards, downwards or rightwards.

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

Solution:

Yes, SARSA on both worlds will converge to the same policy. As long as the exploration policy is $\epsilon-$soft, SARSA is guaranteed to converge to the optimal action value function. The MDP formulation of both the gridworlds lead to effectively the same set of Bellman equations. Since the exploration in both deterministic and stochastic gridworld is effectively $\epsilon-$ greedy, they will converge to the same optimal action value function and hence the same policy.

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.

Contextual bandit formulation:

Suppose we have a $k-$class classification problem with inputs $\{X_1, X_2..., X_n\}$. We also define a mapping $F(X;\theta)$ which is a one-to-one mapping parameterized by θ which takes in an input sample and maps to some output space. A trivial example of such a mapping would be the identity mapping, where $F(X;\theta) = X$.

With this setup, we define a contextual bandit with context s and k arms/actions. When the input is sample X_i, the state $s = F(X_i; \theta)$. If the class corresponding to X_i is y_i, then arm y_i will give a reward of 1, whereas all other arms will give a reward of 0. To summarize,

$$s = F(X_i; \theta)$$

$$R(j|s) = \begin{cases} 1, & \text{if } j = y_i \\ 0, & \text{otherwise} \end{cases}$$

Advantages

- The contextual bandit formulation is inherently robust to an online learning setup. Since the training for one particular state has no dependence whatsoever on other states, training can be done online, whereas it is not so straightforward for the supervised setting.

Disadvantages

- Coming up with the mapping $F(X_i; \theta)$ is not straightforward. Using the identity mapping will make it impossible for the model to generalize as even a small amount of noise will lead to an unseen state for the bandit. A robust mapping $F(X_i; \theta)$ is hence needed for the bandit to actually learn. In fact, most of the effort in the supervised learning approach goes into learning this mapping.
- Given a state s, the rewards of the bandit are not really stochastic in the sense that picking the right label will always yield a reward of 1, whereas a wrong label will yield a reward of 0. In this case, a simple greedy policy is sufficient and formulating it as a bandit problem is rather unnecessary.

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:

Before examining different solution approaches, it is important to clarify the notion of returns and value functions in a non-markovian setting. Whereas in a markov setting, the expected return while computing value functions only involved expectation over

future states, now it also involves an expectation over past states since the return also depends on the history of states now.

TD Learning: The theoretical framework for TD methods require the markovness property since it relies on bootstrapping. The TD target developed is suitable only for MDPs. For non-markov settings, since the value function itself depends on the past sequence of states, the target will keep changing depending on the trajectory taken. While sampling a large number of trajectories may eventually help approximate the expectation over past history, nothing can be said theoretically about convergence in a non-markov setting. Hence, TD Learning is not suitable for non-markov settings.

Monte Carlo methods: MC methods do not require bootstrapping and rely only on sampling. Hence, they do not really require the markovness assumption. By taking sufficient number of samples and averaging them, MC methods should ideally be able to obtain the required value functions even in a non-markov setting (provided we perform the tasks episodically).

Policy Gradient algorithms: The proof of policy gradient theorem assumes markovness and hence it cannot be used as it is for non-markov systems. However, the general idea of optimizing some performance measure based on policy might still be applicable in a broad sense.

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 solution described here is inspired from [1]. The process of generation of appropriate target labels is described below.

We first perform "rollouts" for some arbitrary policy π to compute the action value functions $q_\pi(s, a)$. By rollouts, we essentially mean generating trajectories by following policy π and averaging out the returns to obtain the action value functions.

Suppose we have k possible actions at each state. For each state, we choose the action i for which $q_\pi(s, a_i)$ is maximum and assign i as the label for this state. We may also choose to do a positive-negative labelling by choosing actions which are significantly better as positive samples and actions which are significantly worse as

negative samples. Therefore, finding out a deterministic policy reduces to training a classifier. The classifier can be a standard SVM or a neural network.

This method can be considered as a policy gradient method. Instead of directly parameterizing the policy as is usually done in policy gradient methods, we abstract out the parameterization using a classification network. However, we are internally still parameterizing policies and searching for better policies, thereby making this a policy gradient method. However, note that in some cases such as while using SVMs, we may not be directly computing the gradients for optimization. However, we are still essentially performing policy search.

---

[1] Michail G. Lagoudakis and Ronald Parr. "Reinforcement Learning as Classification: Leveraging Modern Classifiers". In: Proceedings of the Twentieth International Conference on Machine Learning. ICML'03. Washington, DC, USA: AAAI Press, 2003, pp. 424–431. ISBN: 1577351894.