CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, PG **Deadline**: 20 March 2023, 23:59
Name: Aaditya Kumar **Roll number:** EE21D411

- 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: Let $Q(a_i)$ be the expected payoff for pulling arm a_i in the faulty machine. Then, the expected payoff for pulling arm a_i can be expressed as:

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

   where N is the total number of arms.

   The first term in the equation represents the expected payoff for pulling arm a_i and activating arm a_i only, which is equal to the true mean reward $Q^*(a_i)$ weighted by the probability $(1-\epsilon)$ of selecting arm a_i.

   The second term in the equation represents the expected payoff for pulling arm a_i to activate a random arm, which is equal to the average reward $1/N$ of all other arms weighted by the probability ϵ of selecting a random arm.

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: If an action is taken at time t, and the reward received at time $t+\tau$, where τ is the delay, then the return at time t is:

$$G_t = R_{t+\tau} + \gamma R_{t+\tau+1} + \gamma^2 R_{t+\tau+2} + \cdots = \sum_{k=\tau}^{\infty} \gamma^{(k-\tau)} R_{t+k}$$

Since, here we get the reward at time $t+\tau$ instead of time $t+1$ therefore we started the sum from time $t+\tau$.

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

Solution: The TD(0) backup equation for estimating the value function of a given policy in a delayed control task is:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+\tau} + \gamma V(s_{t+\tau}) - V(s_t) \right]$$

Here, since the reward is received at timestep $t + tau$ instead of $t+1$ therefore the update equation is modified to use $V(s_{t+\tau})$ instead of $V(s_{t+1})$

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 $a_1^*(s)$ be the optimal action in M_1, and let $a_2^*(s)$ be the optimal action in M_2 for some state $s \in S$. Then, for every state $s \in S$; we have:

$$a_1^*(s) = \arg\max_{a \in A} Q_1^*(s,a)$$
$$a_2^*(s) = \arg\max_{a \in A} Q_2^*(s,a)$$

Thus, the expression for $a_2^*(s)$ using the given equation for $Q_2^*(s,a)$ is:

$$a_2^*(s) = \arg\max_{a \in A}(Q_1^*(s,a) - f(s))$$

Since, $f(s)$ does not depend on the action a, it does not effect the expression for selecting which action is optimal. Therefore, we can write:

$$a_2^*(s) = \arg\max_{a \in A} Q_1^*(s,a)$$

But this expression is the same as the definition of $a_1^*(s)$. Therefore, we have shown that for every state $s \in S$, $a_1^*(s) = a_2^*(s)$, which means 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: Let Q_1^* and Q_2^* be the optimal state-action value functions for M_1 and M_2, respectively, such that for any $s \in S$ and $a \in A$:

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

By the definition of the state-action value function, we have:

$$Q_1^*(s,a) = \sum_{s'} P(s'|s,a)\left[R_1(s,a,s') + \gamma \max_{a'} Q_1^*(s',a')\right]$$

$$Q_2^*(s,a) = \sum_{s'} P(s'|s,a)\left[R_2(s,a,s') + \gamma \max_{a'} Q_2^*(s',a')\right]$$

From the result of part(a), we know that the optimal policies are the same for both M_1 and M_2.
Thus, substituting $Q_2^*(s,a) = Q_1^*(s,a) - f(s)$ in the state-action value function for $Q_2^*(s,a)$, we have:

$$Q_1^*(s,a) - f(s) = \sum_{s'} P(s'|s,a)\left[R_2(s,a,s') + \gamma \max_{a'}(Q_1^*(s',a') - f(s'))\right]$$

Rearranging and simplifying, we get:

$$Q_1^*(s,a) = \sum_{s'} P(s'|s,a)\left[R_2(s,a,s') + \gamma \max_{a'} Q_1^*(s',a')\right] + f(s) - \gamma \max_{a'} f(s')$$

Comparing this with the expression for $Q_1^*(s,a)$ in terms of R_1 and γ:

$$Q_1^*(s,a) = \sum_{s'} P(s'|s,a) \left[R_1(s,a,s') + \gamma \max_{a'} Q_1^*(s',a') \right]$$

Therefore, R_1 can be expressed in terms of R_2 and f as:

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

and, R_2 can be expressed in terms of R_1 and f as:

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

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 MDP formulation for the above problem is as follows:

State Set: (a_t, b_t) s.t. $a_t \in [0,20]$ and $b_t \in [0,20]$
where, a_t = number of cars at location 1 and b_t = number of cars at location 2 at time step t.

Action Set: max(5,$-b_t$),...,min(5,a_t)
where, a_t = number of cars at location 1 and b_t = number of cars at location

2 at time step t. Here, +ve sign: cars are moved from location 1 to 2 and -ve sign: cars are moved from location 2 to 1.

Reward: At any time step if p_1 and q_1 denote the number of cars that are rented and returned at 1 respectively; and p_2 and q_2 denote the number of cars that are rented and returned at 2 respectively; then:

$r(p_1, q_1, p_2, q_2, (a, b), k) = -2k + 10(min(a - k + q_1, p_1)) + 10(min(b + k + q_2, p_2))$
w.p. $(\frac{3^{p_1}}{p_1!}e^{-3})$ * $(\frac{3^{q_1}}{q_1!}e^{-3})$ * $(\frac{4^{p_2}}{p_2!}e^{-4})$ * $(\frac{2^{q_2}}{q_2!}e^{-2})$.

State Transitions: Suppose we want to go to state $(a^{'}, b^{'})$ from the state (a,b) by shifting k cars, then the probability of transition is denoted by $P((a^{'}, b^{'})|(a, b), k)$
The number of cars in the state $(a^{'}, b^{'})$ are as follows:

$$a' = a - k + q_1 - (min(a - k + q_1, p_1))$$

$$b' = b + k + q_2 - (min(b + k + q_2, p_2))$$

If there are n solutions of the quadruple form to the above equation with i^{th} solution being $(p_1^i, q_1^i, p_2^i, q_2^i)$.
The probability of transition is:
$P((a^{'}, b^{'})|(a, b), k) = \sum_{i=0}^{n}(\frac{3^{p_1^i}}{p_1^i!}e^{-3}) * (\frac{3^{q_1^i}}{q_1^i!}e^{-3}) * (\frac{4^{p_2^i}}{p_2^i!}e^{-4}) * (\frac{2^{q_2^i}}{q_2^i!}e^{-2})$

Discount Factor: The discount factor $\gamma = 0.9$ is used to make sure we are more farsighted and prioritize future 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: An additional reward of -4 must be added to the reward expression, if a second parking lot (independent of how many cars are kept there) is to be used and additionally we can use -2*(k-1) instead of -2*k in the reward

expression to account for whenever Jack's employee shuttles one car to the second location from the first location. This way we can incrementally change our MDP formulation above to account for these nonlinearities.

(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: Since the given transitions is certain, as moving up to 5 cars from one location to other in the night to accommodate for expected sales will move us deterministically from one state to another, we can learn the afterstates i.e the states from which Jack will rent the cars in the morning and proceed instead of using the value function approach.

This will speed up convergence as number of state evaluations is reduced. This can be seen by following example: if the state (a,b) in the evening is (5,5) then the action of moving 3 cars from 1 to 2 will make our morning state (a',b') to be (2,8) and this state is equivalent to evaluation of state action pair of (2,8) in the evening without any movement of cars. There some state action pairs become equivalent.

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: The MDP formulation for the above problem is as follows:

- State Set: {Silence: S, Laughter: L}
- Action Set: {Playing the Organ: O, Burning the Incense: I}
- Reward: +1 for going to S from any state, -1 for going to L from any state.
- State Transition:

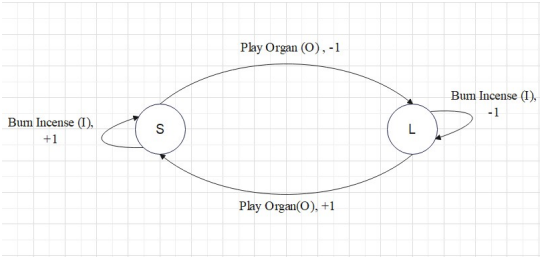

Figure 1: State Transition Diagram for MDP.

- Discount Factor: $\gamma = 0.9$

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

Solution: Let the starting policy π_0 be always burning the incense and not playing the organ i.e,

$$\pi_0 = \{S : I, L : I\}$$

the state can be represented in vector form as $(S\ L)^T$.
The probability transition matrix p_{π_0} is given by

$$p_{\pi_0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and the reward is:

$$r_{\pi_0} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The value function is given by:

$$V_{\pi_0} = \sum_{k=0}^{\infty} \gamma^k p_{\pi_0}^k r_{\pi_0}$$

$$V_{\pi_0} = (I - \gamma p_{\pi_0})^{-1} r_{\pi_0} = \begin{bmatrix} 10 \\ -10 \end{bmatrix}$$

Policy Evaluation Step:

$$\pi_1(S) = \arg\max_a \{O : -1 + 0.9 \times (-10) = -10, I : 1 + 0.9 \times (10) = 10\} = I$$

$$\pi_1(L) = \arg\max_a \{O : 1 + 0.9 \times (10) = 10, I : -1 + 0.9 \times (-10) = -10\} = O$$

Therefore, after the Policy Improvement Step:

$$\pi_1 = \{S : I, L : O\}$$

The probability transition matrix p_{π_1} is given by

$$p_{\pi_1} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

and the reward is:

$$r_{\pi_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The value function is given by:

$$V_{\pi_1} = (I - \gamma p_{\pi_1})^{-1} r_{\pi_1} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

Policy Evaluation Step:

$$\pi_2(S) = \arg\max_a \{O : -1 + 0.9 \times (10) = 8, I : 1 + 0.9 \times (10) = 10\} = I$$

$$\pi_2(L) = \arg\max_a \{O : 1 + 0.9 \times (10) = 10, I : -1 + 0.9 \times (10) = 8\} = O$$

Therefore, after the Policy Improvement Step:

$$\pi_2 = \{S : I, L : O\} = \pi_1$$

This implies that the policy cannot be improved further so the policy π_1 is the optimal policy π^*.

$$\pi^* = \{S : I, L : O\} = \pi_1$$

and the value function for the optimal policy π^* is:

$$V_{\pi^*} = \{S : 10, L : 10\} = V_{\pi_1}$$

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

Solution: My advice to "At Wits End" is if the room is already in state of laughter he should follow the action of playing the organ and then in next time step after reaching the state of quiet he should continue burning incense and not play the organ.

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 policy π' in the stochastic grid world is deterministic and the stochasticity is due to the uncertainty in the environment where an action a taken by agent in accordance to policy π' takes the agent to the next state s' from state s, with probability P(s',s).Therefore, the transition probability is given by:

$$P(s'|s,a) = P(s',s)$$

In the deterministic world the agent has policy π and the transition probability for the agent to go to the next state s' from state s, is given by:

$$P(s'|s,a) = \pi(a|s)$$

For the trajectories to be same for both the worlds, for all states s, s' $\in S$ and action a $\in A(s)$:

$$P(s',s) = \pi(a|s)$$

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

Solution: Yes, SARSA algorithm for both the worlds will converge to the same optimal policy.

Update equation for SARSA:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)))$$

The transition probability i.e the dynamics is same for both the worlds therefore from the Bellman Equation we can say that the state-action value function for state s_t and action a_t will be the same for both the worlds and so will be the SARSA updates.

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 multi-class classification task can be formulated as a contextual bandit problem. In the contextual bandit problem formulation, instead of learning a function that can predict the class label for a given input data point like in the supervised learning setting here the agent is presented with a sequence of data points, and for each data point, the agent must choose an action (i.e., a predicted class label) based on the context (i.e., the features of the data point). After the agent makes a prediction, it receives a reward based on whether its prediction was correct or not. The goal is to maximize the total reward received over the sequence of data points.

For the Contextual Bandit Problem:

- Action: each class label is a separate action
- Context: input features of the data point
- Reward: +1 if the Predicted class label = True label and 0 if both are not same.

Pros of using contextual bandits over supervised learning:

- Since, contextual bandits does not need to be trained on a huge datasets therefore it can be effective when labeled data is scarce as the agent can learn from the reward it receives for taking an action.
- Contextual bandits will perform better if the distribution is changed unlike supervised learning models which performs poorly when the test and train data have different distributions, as Contextual bandits can update its model based on the rewards it gets in the new distribution.
- Since the contextual bandits allow for exploration it may perform better in case of noisy datasets where its difficult to predict the true label.

Cons of using contextual bandits over supervised learning:

- Rewards may not always be so straightforward to define in such cases it would be difficult to use Contextual bandits.
- Contextual bandits may require more iterations to converge than supervised learning, as the learner is updating its model after each data point rather than after a batch of data points.
- In general, supervised learning may be better suited for feature selection than contextual bandits

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: When the state transitions does not only depend on the current state but may also depend on past states the system is said to be non-Markovian.

The suitability of using different solution approaches for such a task, namely, Temporal Difference learning, Monte Carlo methods, and Policy Gradient algorithms are as follows:

- Temporal Difference (TD) learning: TD(0) completely exploits the Markovian property as it updates the next state on the value of the current state. However, TD methods can still work by using an approximation of V(next-state) based on the available information. This approximation can be based on features of the state that are known to be relevant for predicting future outcomes. Meanwhile in the backward view of TD(λ), there is an additional memory variable associated with each state, its eligibility trace. The eligibility trace allows us to get information from previous states and it sits well with the non-Markovian assumption given.

  However, TD methods typically assume that the environment is stationary, which means that the distribution of the rewards and state transitions remains constant over time. If the system is non-stationary, TD methods may fail to converge.

- Monte Carlo (MC) methods: MC methods estimate the value function by sampling complete episodes and using them to update the value function. MC

methods can also work well when the system is not perfectly Markov, as they can learn to account for long-term dependencies in the state transitions. MC methods do not make any assumptions about stationarity, so they can handle non-stationary environments.

However, MC methods require complete episodes to update the value function, which may be impractical or computationally expensive in some settings.

- Policy Gradient (PG) algorithms:In PG algorithms for the proof of Policy Gradient Theorem we assume the Markovian property, as here we have expanded definition of $\pi_\theta(\tau)$ as follows:

$$\pi_\theta(\tau) = P(s_o) \prod_{t=1}^{\tau} \pi_\theta(a_t|s_t) p(s_{t+1}, a_{t+1}|s_t, a_t)$$

Here, we are able to apply the product rule of probability because each new action probability is independent of the previous one (i.e. the Markovian Property). Now that this condition does not hold, the original Policy Gradient Theorem fails and hence this approach is not suitable for Non-Markovian system.

However, one way to incorporate history is to use a recurrent neural network (RNN) as the policy network, which can capture information about past states and actions. The RNN takes the current state and past actions as input, and outputs a probability distribution over actions. The gradient of the policy is then computed using the REINFORCE algorithm, as in standard policy gradient methods.

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: Yes, this method can be considered a policy gradient method because the policy here, is represented as a parametric function as well, such as a neural network, that maps states to actions. The objective is then to optimize the parameters of this function to minimize the difference between the agent's actions and the action corresponding to the true label for the state. This optimization is also done using

gradient descent, where the gradients are computed with respect to the parameters of the policy function.

The appropriate targets for the classifier can be generated by the following approaches:

- We can use a sample-based approach, where the policy is executed in the environment to generate a set of state-action pairs. The state-action pairs can be labeled according to the policy and used as training examples for the classifier.
- Another approach could be applying Imitation learning, in which we use an expert demonstration dataset, where an agent learns to imitate the actions of an expert (possibly a human). In imitation learning, the expert provides a set of demonstrations (i.e., sequences of state-action pairs) to the agent, which then tries to learn a policy that maps states to actions which mimics those like expert's behavior.

The steps for the method that generates appropriate targets for the classifier are as follows:

1. Collect a set of state-action pairs from the environment using sampling method or a dataset from the expert demonstrations.
2. Label each state-action pair with the action taken by the policy.
3. Train a classifier on the labeled state-action pairs to predict the policy labels from states.
4. Test the classifier by using it to predict the policy labels for a new set of states different from the training dataset.
5. Use the predicted policy labels to evaluate the policy and update the policy parameters, if necessary, for improvement.
6. Repeat the steps 1 to 5, till the obtained policy converges to an optimal policy, i.e we get some satisfactory performance.