Written Assignment 1

● Graded

Student

Anirud N

Total Points

31 / 35 pts

Question 1

Question 1

3 / 3 pts

✔ **– 0 pts** Correct

– 1 pt reasoning is correct that for only those target policy we can learn the value function whose state-action pair exists in the optimal policy. But for any arbitrary policy the statement is wrong.

– 2 pts unclear reasoning.

– 3 pts original Q-learning algorithm cannot learn value for any arbitrary policy. Reasoning is completely wrong.

– 2.5 pts incorrect answer and/or reasoning.

– 1 pt partially correct or incomplete reasoning.

– 3 pts incorrect reasoning or not attempted

Question 2

Question 2

1 / 1 pt

✔ **– 0 pts** Correct

– 0.5 pts Incorrect answer

– 0.5 pts Incorrect explanation

Question 3

Question 3

2 / 2 pts

✔ **– 0 pts** Correct

– 1 pt Partial Mistake

– 2 pts Full Wrong

Question 4

Question 4

4 / 4 pts

✔ **– 0 pts** Correct

– 1.3 pts First part Wrong

– 1.3 pts Second part wrong.

– 1.4 pts Third Part Wrong

– 0.6 pts Half a question mistake.

Question 5

Question 5

5 / 5 pts

✔ **– 0 pts** Correct

– 4.5 pts Incorrect

– 3 pts Incomplete explanation for why TD will not work

– 2.5 pts Explain why ?

– 5 pts Did not answer the question which is more suitable in non-markovian environment.

– 0 pts MC do not make any explicit or implicit assumption about markov property. So MC is preferred.

– 1.5 pts TD is not suitable for non-morkovian environment.

– 2.5 pts Explanations are not satisfactory

– 1.5 pts Not clearly explained why MC is suitable

– 1.5 pts Not clearly explained why TD is not suitable

– 0 pts depending upon which PG method you use markov property is required

– 1 pt TD is not suitable for non-markov env.

– 2 pts Non-markov assumption will change the transition dynamics.

– 3.5 pts incorrect explaination

Question 6

Question 6

6 / 6 pts

✔ **– 0 pts** Correct

– 2 pts Two Correct

– 4 pts One Correct

Question 7

Question 7

3 / 3 pts

✔ **– 0 pts** Correct

– 1 pt You need to select the best possible choice for Qa. Ans: dueling DQN

– 0 pts I am assuming you wanted to write Double DQN

– 1 pt You need to select the best possible choice for Qb. Ans: Double DQN

– 1 pt You need to select the best possible choice for Qc. Ans: Expected SARSA

+ 0.5 pts We needed to select the best option for each question. But it also depends on the environment.

– 2 pts Explain why?

+ 1 pt adjusted

Question 8

Question 8

1 / 2 pts

– **0 pts** Correct

– **2 pts** Incorrect

✔ – **1 pt** Partial Marks

Question 9

Question 9

1 / 3 pts

– **0 pts** Correct

– **1 pt** Partial Marks

✔ – **2 pts** Partial Marks

– **3 pts** Incorrect

Question 10

Question 10

5 / 6 pts

– **0 pts** Correct

– **0 pts** Q10.1: linear function approximation as defined is without bias.

https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/idauction2/.g/web/glossary/linear.html

✔ – **1 pt** Q10.1:

linear function is correct:

V(s) = \phi(s) ^T W,

where \phi(s) = [\phi_1(s) , \phi_2(s)] and W = [w_1, w_2]

and with this we can only represent fixed class of function which is of the form:

V(s_3) + 3V(s_1) - 2V(s_2) = 0

If we use linear function approximator, value of any state is fixed once we know the value of other two state.

– **1 pt** Q10.2: if defined with bias, missed bias update.

– **0.5 pts** Q10.2: alpha error

– **0.5 pts** Q10.2: final step unclear

– **1.5 pts** Q10.2: steps not complete

– **1.5 pts** Q10.2: use semi gradient method

– **2.5 pts** Q10.2: incorrect

– **1.5 pts** steps are correct but final answer incorrect

– **0.5 pts** Q10.2 final step small error either in w1 or w2

– **6 pts** not attempted

– **0.5 pts** Q10.1: also show the relation between the value functions:

V(s_3) + 3V(s_1) - 2V(s_2) = 0

– **1.5 pts** see linear function defination:

https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/idauction2/.g/web/glossary/linear.html

Question assigned to the following page: 1

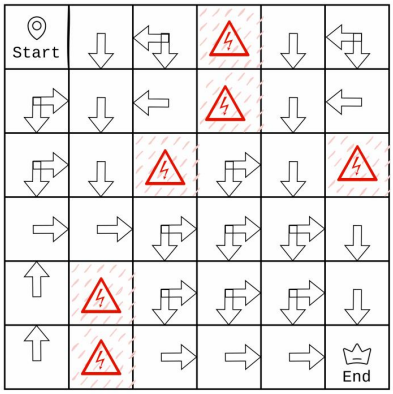
CS6700 : Reinforcement Learning
Written Assignment #1

Topics: Intro, Bandits, MDP, Q-learning, SARSA, FA, DQN **Deadline**: 20/03/2024, 23:55

Name: –Anirud N– **Roll number:** –CE21B014–

- 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. (3 marks) [TD, IS] Consider the following deterministic grid-world.



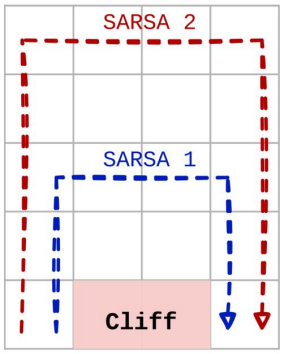
Every actions yields a reward of -1 and landing in the red-danger states yields an additional -5 reward. The optimal policy is represented by the arrows. Now, can you learn a value function of an arbitrary policy while strictly following the optimal policy? Support your claim.

Solution: Off-policy learning is used to learn a Target policy $\pi(a|s)$ while following the deterministic policy $\mu(a|s)$. Off-policy learning has the **Assumption of coverage**: Every action taken under π is also taken, at least occasionally, under μ. In this question, the Optimal policy is the behavior policy - $\mu(a|s)$, and the arbitrary policy that needs to be evaluated is the target policy, $\pi(a|s)$. Every action taken under an arbitrary policy that we wish to evaluate may not always occasionally occur in

Questions assigned to the following page: 1, 3, and 2

the optimal policy. Sometimes, the optimal policy may not contain the actions that belong to the arbitrary policy that we want to evaluate.
Since The assumption of coverage is violated, **we cannot always learn the value function of an arbitary policy by strictly following the optimal policy.**

2. (1 mark) [SARSA] In a 5 x 3 cliff-world two versions of SARSA are trained until convergence. The sole distinction between them lies in the ϵ value utilized in their ϵ-greedy policies. Analyze the acquired optimal paths for each variant and provide a comparison of their ϵ values, providing a justification for your findings.



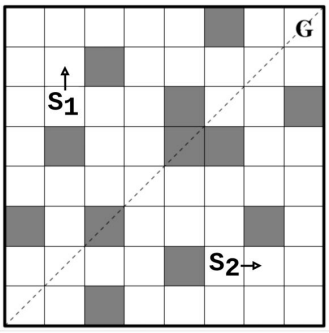
Solution: Higher ϵ means that there is more exploration. Lower epsilon measn lesser exploration. The Sarsa 1 has a lower ϵ value because the agent exploits the known information; it doesn't act much randomly. The agent is more likely to continue with the learned policy. The probability that the agent explores an action apart from the current policy is lesser. So, the agent is more likely to continue to perform the actions that lead to shortest and the most optimal path.
Sarsa 2 has a higher ϵ value because the agent performs a higher exploration. The agent tends to explore random actions rather than the current policy, even if these actions are longer, thus giving us a suboptimal path and solution. Due to higher degree of exploitation, it finds new paths, which are more "safer" ie doesnt suffer from higher negative rewards that easily as the cliff is far away but longer in terms of number of steps.
Note that Sarsa gives the optimal policy keeping "exploration" in mind

$$\epsilon_2 > \epsilon_1$$

3. (2 marks) [SARSA] The following grid-world is symmetric along the dotted diagonal. Now, there exists a symmetry function $F : S \times A \to S \times A$, which maps a state-action

Question assigned to the following page: 3

pair to its symmetric equivalent. For instance, the states S_1 and S_2 are symmetrical and F(S_1,North) = S_2,East.



Given the standard SARSA pseudo-code below, how can the pseudo-code be adapted to incorporate the symmetry function F for efficient learning?

Algorithm 1 SARSA Algorithm

Initialize Q-values for all state-action pairs arbitrarily
for each episode **do**
  Initialize state s
  Choose action a using ϵ-greedy policy based on Q-values
  while not terminal state **do**
    Take action a, observe reward r and new state s'
    Choose action a' using ϵ-greedy policy based on Q-values for state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma Q(s',a') - Q(s,a)\right)$
    $s \leftarrow s'$, $a \leftarrow a'$
  end while
end for

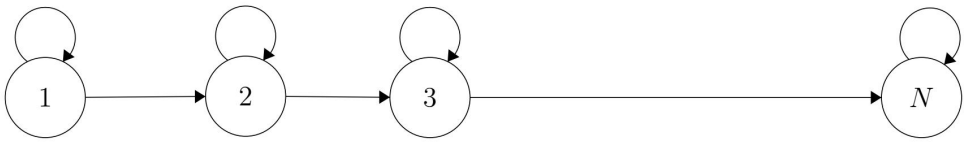
Solution:

Questions assigned to the following page: 4 and 3

Algorithm 2 SARSA Algorithm

Initialize Q-values for all state-action pairs arbitrarily
for each episode **do**
  Initialize state s
  Choose action a using ϵ-greedy policy based on Q-values
  while not terminal state **do**
    Take action a, observe reward r and new state s'
    Choose action a' using ϵ-greedy policy based on Q-values for state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma Q(s',a') - Q(s,a)\right)$
    $s_{symmetric}, a_{symmetric} = F(s,a)$
    $s'_{symmetric}, a'_{symmetric} = F(s',a')$
    $Q(s_{symmetric}, a_{symmetric}) \leftarrow Q(s_{symmetric}, a_{symmetric}) + \alpha\left(r + \gamma Q(s'_{symmetric}, a'_{symmetric}) - Q(s_{symmetric}, a_{symmetric})\right)$
    $s \leftarrow s'$, $a \leftarrow a'$
  end while
end for

4. (4 marks) [VI] Consider the below deterministic MDP with N states. At each state there are two possible actions, each of which deterministically either takes you to the next state or leaves you in the same state. The initial state is 1 and consider a shortest path setup to state N (Reward -1 for all transitions except when terminal state N is reached).



Now on applying the following Value Iteration algorithm on this MDP, answer the below questions:

Algorithm 3 Value Iteration Algorithm

1: Initialize V-values for all states arbitrarily over the state space S of N states. Define a permutation function ϕ over the state space
2: $i \leftarrow 0$
3: **while** NotOptimal(V) **do**
4: $\quad s \leftarrow \phi(i \mod N)$
5: $\quad V(s) \leftarrow \max_a \sum_{s,r'} p(s', r|s, a)[r + \gamma * V(s')]$
6: $\quad i \leftarrow i + 1$
7: **end while**

Question assigned to the following page: 4

1. (1 mark) Design a permutation function ϕ (which is a one-one mapping from the state space to itself, defined here), such that the VI algorithm converges the fastest and reason about how many steps (value of i) it would take.

   Solution: Set $\phi(i \mod N+1)$ **to** $N - i \mod N$
   $V(N) = 0$
   Let's assume a "good initialization," i.e., where all the state values V(S) is negative for $s < 0$. In this case, the V(N-1) gets updates to 0, V(N-2) to -1, V(N-3) to -2... and V(1) to -N+1. These update values from backward. So, a total of N steps(each step for each state) need to be updated.
   So for a "good initialization," **we require only N steps to converge to an optimal V.**
   So, we eventually backtracked from the value of the last state to all states.
   But not always we get such an initialisation. The V(S) values can have significant positive values for all states S except N. In this case, the difference between the current state V(s) and the target optimal state V(s+1) is positive, and so it chooses the action that confines the agent to reach the same state s again instead of s+1.
   It takes at least N^2 iterations to reduce these High positive initialized values and later N steps to converge to the optimal policy. So, we can say that in the worst case, it takes $O(n^2)$ to converge.

2. (1 mark) Design a permutation function ϕ such that the VI algorithm would take the most number of steps to converge to the optimal solution and again reason how many steps that would be.

   Solution: Set $\phi(i \mod N+1)$ **to** $i \mod N+1$
   This leads to a cyclic updation of 1 to N and back to 1
   Assuming a "good initialization," i.e., when the values of states are lesser, i.e., V(s) is at least less than 0.
   In this case, starting from 1 to N, when we reach state N-1, we update V(N-1) to its optimal value. In the second cycle, we update N-2, then N-3, and so on..
   So For all the updates to happen, for a "good initialization", **it takes O(n^2) iterations to update to optimal policy. Note that a good initialization means:** At any state S [1 to N-2]:

   $$\gamma * V(S) - 1 < \gamma * V(S+1) - 1$$

   If this does not hold true, then initially, updates will take place until this condition is satisfied.

3. (2 marks) Finally, in a realistic setting, there is often no known semantic meaning

Questions assigned to the following page: 4 and 5

associated with the numbering over the sets and a common strategy is to randomly sample a state from S every timestep. Performing the above algorithm with s being a randomly sampled state, what is the expected number of steps the algorithm would take to converge?

Solution: Assuming a "good initialization," i.e., when the values of states are lesser, i.e., V(s) is at least less than 0. Let $P(s)$ be the probability of selecting one state.

$$P(s) = \frac{1}{N}$$

For updating a state S = N-1 to its optimal value, the expected number of steps is N. Then For updating a state S = N-2 to its optimal value, the expected number of steps is N. This goes on for all states

Expected Total number of steps for convergence is of the order = $\sum_1^{N-1} N = N(N-1)$

If the initialization is bad, i.e., if the V(s) values are very large, then it takes initial steps to update and reduce the V(s) values to smaller values (or values lesser than 0). For instance, let's say that V(N-1) has a value $K > 0$. Then, at first update, V(N-1) reduces to K -1. The second update is K-2.. and at t the update, it will be K-t. So after K updates, V(s) has a value of 0, and here it chooses the optimal action as going to state N. So it takes K+1 updates on state N-1 to reach optimal policy at state N-1. Similarly, at a state N-p, if the value function has a value V(N-p) =K initialised, then it takes K+p updates on N-p state to converge to optimal policy at state N-p

Note: Do not worry about exact constants/one-off differences, as long as the asymptotic solution is correct with the right reasoning, full marks will be given.

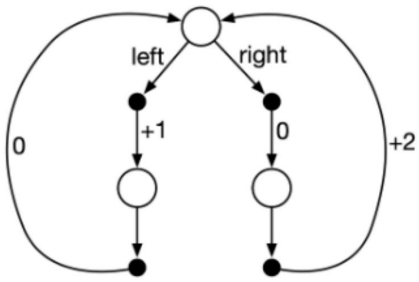
5. (5 marks) [TD, MC] Suppose that the system that you are trying to learn about (estimation or control) is not perfectly Markov. Comment on the suitability of using different solution approaches for such a task, namely, Temporal Difference learning, Monte Carlo methods. Explicitly state any assumptions that you are making.

Solution: Assuming that the system is not perfectly Markov(violates Markov property), i.e., the future state might depend on a broader context, including past states and actions
Both TD Learning and MC control can be model free. They both sample. But TD also bootstraps wheras MC doesnt bootstrap
Monte Carlo : Monte Carlo methods estimate value functions by averaging sampled returns. It is episodic in nature, i.e., they traverse throughout the episode. Only

Questions assigned to the following page: 5 and 6

after the episode ends, value function updates are updated.
Since they rely only on full episodes, they handle Non- Markovian systems better. By sampling complete episodes, they explore different trajectories. In case of instances where the rewards are delayed, ie the actual representation of the reward for taking a state s occurs after a long time, MC control proves to be a better method as it takes into acc all the rewards from the trajectory. Monte Carlo estimates may have high variance due to limited samples. They can be computationally expensive
TD Control: TD(n) takes into account and samples the n steps or states following the current states and uses it to update the value function. It doesn't necessarily wait for full episodes to complete to update the value function. TD does bootstrapping; the updates are based on the states' value function estimates. In the case of TD(0), immediate rewards and the value function estimates of the following state are looked into to update the value function at the current state. So, TD(0) suffers in Non-Markovian nature of the system. The higher the value of n in TD(n), the more sampling is done until n timesteps follow the current state and the value function update is done.As n increases, the effect of Non-Markovian nature of the system is reduced. Note that as n approaces to the length of episode, the TD approaches to MC.

6. (6 marks) [MDP] Consider the continuing MDP shown below. The only decision to be made is that in the *top* state (say, s_0), where two actions are available, *left* and *right*. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. Calculate and show which policy will be the optimal:



(a) (2 marks) if $\gamma = 0$

Solution:

$$V_\pi(s) = \sum \pi(a|s) \sum p(s', r'|s, a)[r' + \gamma * V_\pi(s)]$$

Assume that left is denoted by l and right is denoted by r. The state that we reach upon taking a left is S_l, and the state upon taking a right is S_r

Questions assigned to the following page: 6 and 7

$$V_\pi(S_l) = 0$$
$$V_\pi(S_r) = 2$$
$$V_{\pi_l}(s) = 1 + \gamma * V_\pi(S_l)$$
$$V_{\pi_r}(s) = 0 + \gamma * V_\pi(S_r)$$
$$V_{\pi_l}(s) = 1$$
$$V_{\pi_r}(s) = 2 * \gamma$$

if γ is 0 then

$$V_{\pi_l}(s) > V_{\pi_r}(s)$$
$$(1 > 0)$$

Going Left is an optimal Policy

(b) (2 marks) if $\gamma = 0.9$

Solution: if γ is 0.9 then

$$V_{\pi_l}(s) < V_{\pi_r}(s)$$
$$(1 < 1.8)$$

Going Right is an optimal Policy

(c) (2 marks) if $\gamma = 0.5$

Solution: if γ is 0.5 then

$$V_{\pi_l}(s) = V_{\pi_r}(s)$$
$$(1 = 1)$$

Going Left and Right is an optimal Policy as they are equally bad due to equal value functions values

7. (3 marks) Recall the three advanced value-based methods we studied in class: Double DQN, Dueling DQN, Expected SARSA. While solving some RL tasks, you encounter the problems given below. Which advanced value-based method would you use to overcome it and why? Give one or two lines of explanation for 'why'.

   (a) (1 mark) Problem 1: In most states of the environment, choice of action doesn't matter.

Question assigned to the following page: 7

Solution: Dueling DQN
The advantage function is defined as

$$A(s,a) = V_\pi(s) - Q(s,a)$$

so there are two networks used to calculate $V_\pi(s)$ and $Q(s,a)$, independently. and used to finally calculate Advantage function.
In states with little effect from individual actions, the value network can concentrate on predicting the state value (V(s)) independently of those actions.
The advantage network helps capture the differences between action values (A(s, a)).
Dueling DQN focus exploration on states where actions matter, leading to more efficient learning.

(b) (1 mark) Problem 2: Agent seems to be consistently picking sub-optimal actions during exploitation.

Solution: Double DQN
While this can occur due to various reasons, one possible reason is the "Maximisation Bias,"- which occurs when one action is being chosen with a higher probability than the other action due to a lesser number of observations. DDQN uses different functions to evaluate and select policy action.

(c) (1 mark) Problem 3: Environment is stochastic with high negative reward and low positive reward, like in cliff-walking.

Solution: Expected SARSA
The normal sarsa update equation:

$$Q(s_t,a_t) < -Q(s_t,a_t) + \alpha(r_{t+1} + \gamma * Q(s_{t+1},a_{t+1} - Q(s_t,a_t)$$

a_{t+1} introduces variance and slows down convergence, especially when the environment is stochastic with high -ve reward and low +ve reward.
Using the expectation value of Q reduces the variance in the update.

$$Q(s_t,a_t) < -Q(s_t,a_t) + \alpha(r_{t+1} + \gamma * E_\pi[Q(s_{t+1},a_{t+1}] - Q(s_t,a_t)$$

Questions assigned to the following page: 9 and 8

8. (2 marks) [REINFORCE] Recall the update equation for *preference* $H_t(a)$ for all arms.

$$H_{t+1}(a) = \begin{cases} H_t(a) + \alpha\,(R_t - K)\,(1 - \pi_t(a)) & \text{if } a = A_t \\ H_t(a) - \alpha\,(R_t - K)\,\pi_t(a) & \text{if } a \neq A_t \end{cases}$$

where $\pi_t(a) = e^{H_t(a)} / \sum_b e^{H_t(b)}$. Here, the quantity K is chosen to be $\bar{R}_t = \left(\sum_{s=1}^{t-1} R_s\right)/t - 1$ because it empirically works. Provide concise explanations to these following questions. Assume all the rewards are non-negative.

(a) (1 mark) How would the quantities $\{\pi_t(a)\}_{a \in \mathcal{A}}$ be affected if K is chosen to be a large positive scalar? Describe the policy it converges to.

> **Solution: Explorative - and converges to equal probabilities ie the policy converges to a more uniform exploration strategy**.
> Reason:
> If K is a large positive scalar, then H(a) value decreases each time the action a is chosen and increases for all other non chosen actions. Therefore, each time an action is chosen, the probability of choosing the action again, i.e., $\pi(a)$, is reduced as H(a) is reduced and the $\pi(a')$ for all other non-selected actions a' increases.

(b) (1 mark) How would the quantities $\{\pi_t(a)\}_{a \in \mathcal{A}}$ be affected if K is chosen to be a small positive scalar? Describe the policy it converges to.

> **Solution: Exploitative and greedy**
> Reason:
> if K is a small positive value, then if an action a_t is chosen at time t, the $H(a_t)$ value increases as $R_t - K$ is still positive. So, the $\pi(a)$ increases for chosen action a_t and decreases for all other non-chosen actions.

9. (3 marks) [Delayed Bandit Feedback] Provide pseudocodes for the following MAB problems. Assume all arm rewards are gaussian.

(a) (1 mark) UCB algorithm for a stochastic MAB setting with arms indexed from 0 to $K - 1$ where $K \in \mathbb{Z}^+$.

> **Solution:**

Question assigned to the following page: 9

Algorithm 4 UCB

Initialize: Play each arm once, h = exploration parameter
for $t = 1$ **to** $TotalTimeSteps$ **do**
    for $j = 0$ **to** $K - 1$ **do** ▷ Loop across K arms
        Calculate UCB value for the arm j:
        $UCB_j \leftarrow \hat{\mu}_j + \sqrt{\frac{h \cdot \ln(t)}{N_j}}$
        $t \leftarrow$ total number of time steps happened so far
        $\hat{\mu}_j \leftarrow$ mean reward of arm j
        $N_j \leftarrow$ number of times arm j has been played
    end for
    Arm with maximum value of UCB is selected and performed:
    $A_t \leftarrow \arg\max(UCB_j)$
    Perform the action A_t
    reward r_t is observed
    Update mean reward and number of times arm is played for arm A_t:

    $\hat{\mu}_A \leftarrow \frac{N_A \cdot \hat{\mu}_A + r_t}{N_A + 1}$
    $N_A \leftarrow N_A + 1$

end for

(b) (2 marks) Modify the above algorithm so that it adapts to the setting where agent observes a feedback tuple instead of reward at each timestep. The feedback tuple h_t is of the form $(t', r_{t'})$ where $t' \sim \text{Unif}(\max(t - m + 1, 1), t)$, $m \in \mathbb{Z}^+$ is a constant, and $r_{t'}$ represents the reward obtained from the arm pulled at timestep t'.

Solution:

Questions assigned to the following page: 9 and 10

Algorithm 5 Modified ALGORITHM UCB

Initialize: Play each arm once, h = exploration parameter
for $t = 1$ **to** $TotalTimeSteps$ **do**
  for $j = 0$ **to** $K - 1$ **do** ▷ Loop across K arms
    Calculate UCB value for the arm j:
    $UCB_j \leftarrow \hat{\mu}_j + \sqrt{\frac{h \cdot \ln(t)}{N_j}}$
    $t \leftarrow$ total number of time steps happened so far
    $\hat{\mu}_j \leftarrow$ mean reward of arm j
    $N_j \leftarrow$ number of times arm j has been played
  end for
  Arm with maximum value of UCB is selected and performed:
  $A_t \leftarrow \arg\max(UCB_j)$
  Perform the action A_t
  Sample feedback tuple:
  $t' \sim \text{Unif}(\max(t - m + 1, 1), t)$
  $h_t = (t', r_{t'})$
  $r_{t'}$ is the reward obtained from arm pulled at timestep t'

  Update mean reward and number of times arm is played for arm A_t:

  $\hat{\mu}_A \leftarrow \frac{N_A \cdot \hat{\mu}_A + r_{t'}}{N_A + 1}$
  $N_A \leftarrow N_A + 1$

end for

10. (6 marks) [Function Approximation] Your are given an MDP, with states s_1, s_2, s_3 and actions a_1 and a_2. Suppose the states s are represented by two features, $\Phi_1(s)$ and $\Phi_2(s)$, where $\Phi_1(s_1) = 2$, $\Phi_1(s_2) = 4$, $\Phi_1(s_3) = 2$, $\Phi_2(s_1) = -1$, $\Phi_2(s_2) = 0$ and $\Phi_2(s_3) = 3$.

    1. (3 marks) What class of state value functions can be represented using only these features in a linear function approximator? Explain your answer.

Solution: Let $w = [w1, w2]^T$ be the feature weights for ϕ_1 and ϕ_2 respectively

$$\phi(s) = [\phi_1(s), \phi_2(s)]$$

The Value function for each state that maps state space to real numbers can be given by a linear combination of features of the space, ie the entries of $\phi(s)$

$$V(s) = \phi(s)^T \cdot w$$

Question assigned to the following page: 10

Simplifying gives us

$$V(s) = w1 * \phi_1(s) + w2 * \phi_2(s)$$

2. (3 marks) Updated parameter weights using gradient descent TD(0) for experience given by: $s_2, a_1, -5, s_1$. Assume state-value function is approximated using linear function with initial parameters weights set to zero and learning rate 0.1.

Solution:

$$V(s) = w1 * \phi_1(s) + w2 * \phi_2(s)$$

$$\delta_t = r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)$$

inital weights are set to 0. So the Value functions are also initialized to 0 for all s
for all s,$V(s)$ is 0
so plugging in the values r_{t+1} as -5 and $V(s_1)$ and $V(s_2)$ as 0
we get δ_t as -5

$$\phi(s_2) = [\phi_1(s_2), \phi_2(s_2)]^T$$

$$w_{t+1} = w_t + \alpha * \delta_t \cdot \phi(s_2)$$

$$w_{t+1} = 0 + 0.1 * (-5) \cdot [4, 0]^T$$

$$w1 = -2$$

$$w2 = 0$$

$$w_{t+1} = [-2, 0]^T$$