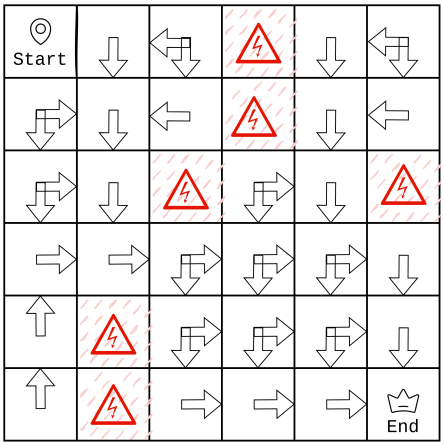
CS6700 : Reinforcement Learning
Written Assignment #1

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

Name: –Shuvrajeet Das– **Roll number:** –CS23E001 –

- 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.**

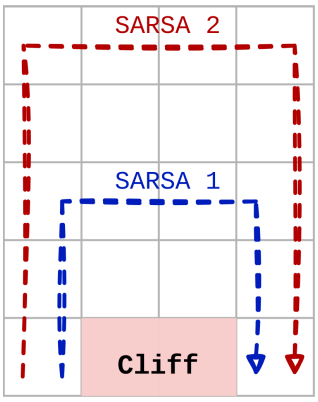
1. (3 marks) [TD, IS] Consider the following deterministic grid-world.



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.

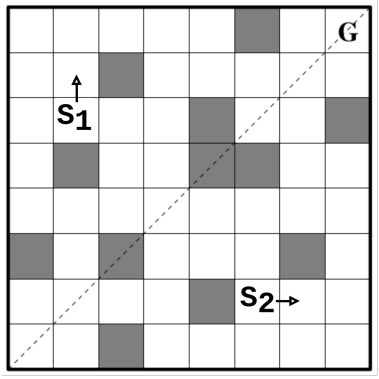
Solution: The value function estimates the expected return of following a particular policy, when strictly following the optimal policy, the value function will converge to the true value function associated with the optimal policy. This is because the updates are based on experiences obtained by following the optimal actions, ensuring that the value estimates reflect the optimal policy's behaviour.

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: The SARSA 1 has a lower ϵ for an ϵ-greedy policy which tends to make it more greedy enabling it to explore less and probing it to choose more unsafe actions compared to SARSA 2 having a higher value of epsilon.

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

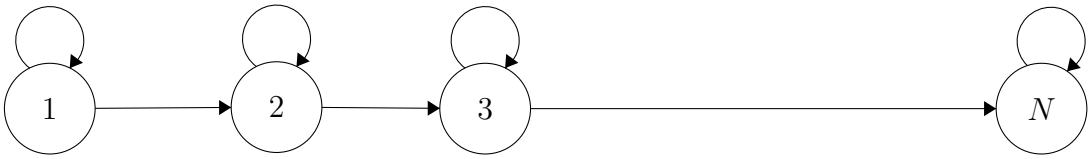
Initialize Q-values for all state-action pairs arbitrarily
for each episode **do**
  Initialize state s
  Choose action a using ϵ-greedy policy based on Q-values
  while not terminal state **do**
    Take action a, observe reward r and new state s'
    Choose action a' using ϵ-greedy policy based on 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:

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'
    $s_{sym}, a_{sym} \leftarrow F(s,a)$
    $s'_{sym}, a'_{sym} \leftarrow F(s',a')$
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma Q(s',a') - Q(s,a)\right)$
    $Q(s_{sym},a_{sym}) \leftarrow Q(s_{sym},a_{sym}) + \alpha\left(r + \gamma Q(s'_{sym},a'_{sym}) - Q(s_{sym},a_{sym})\right)$
    $s \leftarrow s'$, $a \leftarrow a'$
    $s_{sym} \leftarrow s'_{sym}$, $a_{sym} \leftarrow a'_{sym}$
  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

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

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: The permutation function can be $\phi(i) = N - (i \mod N)$. It would take at least 1 step to reach the goal considering the best-state scenario.

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: The permutation function can be $\phi(i) = (i+1) mod N$. It would take at least N step to reach the goal considering the best-state scenario.

3. (2 marks) Finally, in a realistic setting, there is often no known semantic meaning 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: The expected number of steps for the algorithm to converge is N

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.

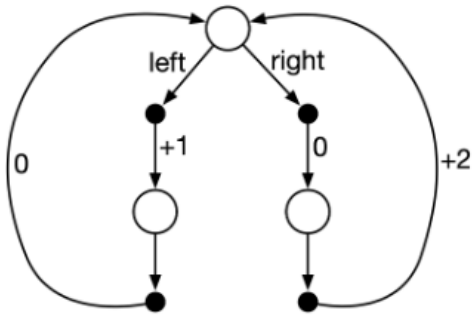
Solution: Temporal Difference (TD) Learning:

- Suitable for partially observable environments.
- Updates estimates incrementally after each transition.
- Assumes partial Markov property.
- Efficient in terms of computation and memory.

Monte Carlo (MC) Methods:

- Suitable for completely non-Markovian environments.
- Requires complete episodes for accurate estimates.
- Robust to violations of the Markov property.
- Less efficient, especially with long episodes or challenging exploration.

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: $V_\pi = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$
for $\pi_{left} = 1 + \mathbb{E}_{\pi_{left}}[0 \times G_{t+1} | S_t = s] = 1$
for $\pi_{right} = 0 + \mathbb{E}_{\pi_{right}}[0 \times G_{t+1} | S_t = s] = 0$
thus π_{left} is best.

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

Solution: $V_\pi = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$
for $\pi_{left} = 1 + \mathbb{E}_{\pi_{left}}[0.9 \times G_{t+1} | S_t = s] = 1 + 0.9 \times 0 = 1$
for $\pi_{right} = 0 + \mathbb{E}_{\pi_{right}}[0.9 \times G_{t+1} | S_t = s] = 0 + 0.9 \times 2 = 1.8$
thus π_{right} is best

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

Solution: $V_\pi = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s]$
for $\pi_{left} = 1 + \mathbb{E}_{\pi_{left}}[0.9 \times G_{t+1} | S_t = s] = 1 + 0.5 \times 0 = 1$
for $\pi_{right} = 0 + \mathbb{E}_{\pi_{right}}[0.9 \times G_{t+1} | S_t = s] = 0 + 0.5 \times 2 = 1$
thus bot policies are optimal

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.

Solution: Duelling DQN separates the value estimation into state value and advantage value, which helps in situations where actions have little impact on the value function. This architecture allows the agent to focus on learning the value of states independently of the specific actions available.

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

Solution: Double DQN would be appropriate to address this issue. By decoupling the action selection and value estimation, Double DQN reduces overestimation bias, leading to more accurate action-value estimates. This helps in selecting optimal actions during exploitation more reliably.

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

Solution: Expected SARSA calculates the expected value of the next state-action pair, which is particularly useful in stochastic environments with high variance in rewards. By considering the expected value, the agent can better navigate the environment, especially when dealing with high negative and low positive rewards, such as in cliff-walking, where one wrong move can lead to significant penalties.

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: If a=A_t then because of the larger value of K, the term $(R_t - K)$ will be more negative, assuming positive feedback then $H_{t+1}(a)$ will decrease. Thus the policies will not accept actions with high $H_t(a)$ values and vice versa, making it converge towards as greedy policy.

(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: If a=A_t then because of the smaller value of K the term $(R_t - K)$ will be close to Rt (assuming non-negative rewards). As a result, the preference $H_{t+1}(a)$ will increase if the chosen action A_t leads to a positive reward. The policy will favor actions that yield positive rewards and for the negation the

preference $H_{t+1}(a)$ will increase for actions other than the chosen action. As K becomes smaller, the policy will converge toward an exploratory policy.

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: Input: Number of arms K
Initialization:

- Initialize empirical means for each arm: $Q[a] = 0$, $N[a] = 0$ for all arms a
- Initialize time step $t = 0$

While True:

- Select arm a_t according to:

$$a_t = \arg\max_a Q[a] + \sqrt{\frac{2\log(t+1)}{N[a]}}$$

- Observe reward r_t from arm a_t
- Update empirical mean and counts:

$$N[a_t] += 1$$

$$Q[a_t] = Q[a_t] + \frac{1}{N[a_t]} \cdot (r_t - Q[a_t])$$

- Increment time step t

   (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: Input: Number of arms K, parameter m
Initialization:

- Initialize empirical means for each arm: $Q[a] = 0$, $N[a] = 0$ for all arms a
- Initialize time step $t = 0$

While True:

- Sample a time step t' uniformly from the interval $[\max(t - m + 1, 1), t]$
- Select arm a_t according to:

$$a_t = \arg\max_a Q[a] + \sqrt{\frac{2\log(t+1)}{N[a]}}$$

- Observe feedback tuple (t', r'_t) from arm a_t
- Update empirical mean and counts:

$$N[a_t] += 1$$

$$Q[a_t] = Q[a_t] + \frac{1}{N[a_t]} \cdot (r_{t'} - Q[a_t])$$

- Increment time step t

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: The function $V(s)$ can be defined as $V(s) = w_1 * \Phi_1(s) + w_2 * \Phi_2(s)$, where w_1 and w_2 can be defined as the weights of the assigned to each feature.

    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: The gradient of $V(s)$ becomes $\nabla V(s) = [\Phi_1(s), \Phi_2(s)]$
    also, $\Delta w_i = \alpha * (r + \gamma * V(s') - V(s)) * \Phi_i(s)$
    given $V(s) = 0$ initially
    The update for each of the weight becomes

$\Delta w_1 = 0.1 * (-5 + 1 * V(s_1)) * \Phi_1(s_2) = 0.1 * (-5 + V(s_1)) * 4$
$\Delta w_2 = 0.1 * (-5 + 1 * V(s_1)) * \Phi_2(s_2) = 0.1 * (-5 + V(s_1)) * 0$
since $\Phi_2(S_2) = 0$ then update for $w_2 = 0$