ECE 2795: Reinforcement Learning
Homework assignment 1

Due September 25, 2023

1 T-step state probabilities (20 Pt)

1.1

Consider a Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where $\mathcal{S}$ and $\mathcal{A}$ are discrete state and action spaces with dimensions N and M, respectively. Write down the probability of being at state s' after T steps when starting at state s, when following a policy $\pi(a|s)$. This is

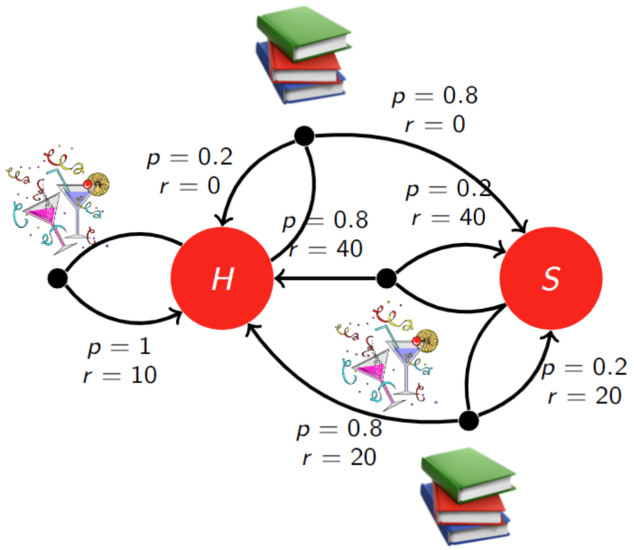
$$P(S_T = s' | S_0 = s), \tag{1}$$

where S_T and S_0 are random variables that represent the state at times T and 0.

Hint: Write it for cases $T = 1$ and $T = 2$, then use induction. This problem is a generalization to T steps of the one and two transitions (slides 25-26).

1.2

In the following parts, we will examine the Happy-Sad example from slide 32.



If the student flips a balanced coin to take each decision, compute the probability of the trajectory $\tau_3 = (happy, books, happy, books, sad, party, sad)$, that is

P($\mathrm{T}_3 = \tau_3$)
$= P(S_3 = \texttt{happy}, A_2 = \texttt{books}, S_2 = \texttt{happy}, A_1 = \texttt{books}, S_1 = \texttt{sad}, A_0 = \texttt{party}, S_0 = \texttt{sad})$

1.3

Assume there are two types of students, or policies: those who always study and those who always party.

Write a code to compute the probability of being in some state s' after T time steps when starting from state s at time $t = 0$, using either of the two policies.

Then, write a script to compute the following probabilities for both students:

1. $P(S_1 = \texttt{happy} \mid S_0 = \texttt{happy})$
2. $P(S_1 = \texttt{happy} \mid S_0 = \texttt{sad})$
3. $P(S_2 = \texttt{happy} \mid S_0 = \texttt{happy})$
4. $P(S_2 = \texttt{happy} \mid S_0 = \texttt{sad})$
5. $P(S_3 = \texttt{happy} \mid S_0 = \texttt{happy})$
6. $P(S_3 = \texttt{happy} \mid S_0 = \texttt{sad})$

Refer to the attached Jupyter Notebook for a starter code that provides the transition probabilities for the MDP.

Compute $P(S_2 = \texttt{happy} \mid S_0 = \texttt{sad})$ for one of the students using the expression derived in 1.1 and compare it to the output of your script.

2 Policy Improvement Theorem (20 Pt)

Given an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$ with discount rate γ and a policy $\pi(a|s)$ we define the q function as

$$q_\pi(s, a) = E[R_t + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]. \tag{2}$$

Let us construct a new policy, $\pi'(a|s)$ such that for any state $s \in \mathcal{S}$ it selects $a =_{a \in \mathcal{A}} q(s, a)$ once and then it follows $\pi(a|s)$. The Policy Improvement Theorem then says that

$$v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}. \tag{3}$$

We will now prove this over several steps. The basic idea of the proof is that we consider starting at any arbitrary state $s \in \mathcal{S}$ and taking one step using $\pi'(a|s)$ and following $\pi(a|s)$ afterward. This should yield a larger expected return. Next, we consider taking two steps with $\pi'(a|s)$ and following $\pi(a|s)$ afterward, and so on.

2.1

Show that $E_{a \sim \pi'}[q_\pi(s, a)] \geq v_\pi(s)$. This tells us that the average reward of taking just one step with $\pi'(a|s)$ and then following $\pi(a|s)$ must be at least as good as if we just followed the policy $\pi(a|s)$ the whole time.

2.2

We define for convenience $Q_1 := E_{a \sim \pi'}[q_\pi(s, a)]$. Now, consider $\pi'(a|s)$ such that we apply $a =_{a \in \mathcal{A}} q(s, a)$ twice and then we follow $\pi(a|s)$.

Show that $Q_2 := E_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2})] \geq Q_1$.

2.3

Defining similarly Q_T for all $t \geq 1$ and using similar steps to 2.2 show that $Q_{T+1} \geq Q_T$.

Using this we can reason that $Q_\infty \geq v_\pi(s)$. What is Q_∞?

(Hint: Look at the form that Q_2 takes from 2.2)

The full statement of the Policy Improvement Theorem is actually: Given two policies, $\pi(a|s)$ and $\pi'(a|s)$, if $E_{a\sim\pi'}[q_\pi(s,a)] \geq v_\pi(s), \forall s \in \mathcal{S}$, then $v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}$.

This is a broader statement because you can construct π' through any means. For example, instead of each picking the best action value at a single state, you can do it for all states! And our proof still works.

3 Policy Iteration (20 Pt)

Policy Iteration is an algorithm that takes advantage of the Policy Improvement Theorem. The basic idea is to start with a policy π_0, run policy evaluation to compute $v_{\pi_0}(s), \forall s \in \mathcal{S}$. Then create π_1 by taking the best action at each state and evaluate $v_{\pi_1}(s), \forall s \in \mathcal{S}$ and repeat to generate $\pi_2, \pi_3, \ldots$ until π converges.

Given the Policy Improvement Theorem, this method of generating new policies π is guaranteed to strictly improve the value function for some states unless the original policy is already optimal.

The algorithm is summarized below. For additional details, refer to Page 80 of Reinforcement Learning by Sutton and Barto, 2019.

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Implement Policy iteration for the Happy-Sad Markov Decision Process of Exercise 1 starting with a uniform policy for both states. Use the starter code provided in the attached Jupyter Notebook. It already includes the rewards and transition probabilities of the MDP. Provide three plots:

1. State value $V(s)$ of each state (happy and sad) versus the iteration step number (the number of times policy evaluation, step 2 in the pseudo-code, is executed).
2. Table of the policy $\pi(a|s)$ in each state after convergence.
3. Table with $v(s)$ upon convergence.

Can you think of any problems with this method?

4 VALUE ITERATION (20 PT)

To address the inefficiencies of Policy Iteration, we can eliminate the need to fully evaluate each policy. Instead, we can interleave the value function updates and policy updates. This algorithm is called Value Iteration and additional details can be found on Page 82 of Reinforcement Learning by Sutton and Barto, 2019.

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \text{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Implement Value Iteration for the Happy-Sad Markov Decision Process of Exercise 1 starting with a uniform policy for both states. Use the starter code in the attached Jupyter Notebook. It already includes the rewards and transition probabilities of the MDP. Provide three plots:

1. State value $V(s)$ of each state (happy and sad) versus the iteration step number.
2. Table of the policy $\pi(a|s)$ in each state after convergence.
3. Table with $v(s)$ upon convergence.

5 BELLMAN EQUATIONS (20 PT)

5.1

Write the Bellman equations for the student model in problem 1.

5.2

Verify that the solutions in parts 3 and 4 satisfy the Bellman equations for this MDP.