Programming Assignment 3: (Reinforcement Learning) EE-5180

April 6, 2024

1. Submission deadline: Monday (15/04/2024), 10.00 PM
2. Working code has to be submitted with proper comments and it should not be copied from your friends.
3. Working of code has to be understood and have to write independently.
4. Note:
   - Programming assignment mean to improve your programming skill and to develop your ability to programs in machine learning problems.
   - If found that code is copied, there will be negative penalty for this.
5. This assignment has two parts. Part A is on Markov decision processes and algorithm. Part B will be on Q-learning algorithm.

1 Part A: Markov decision processes and Algorithms

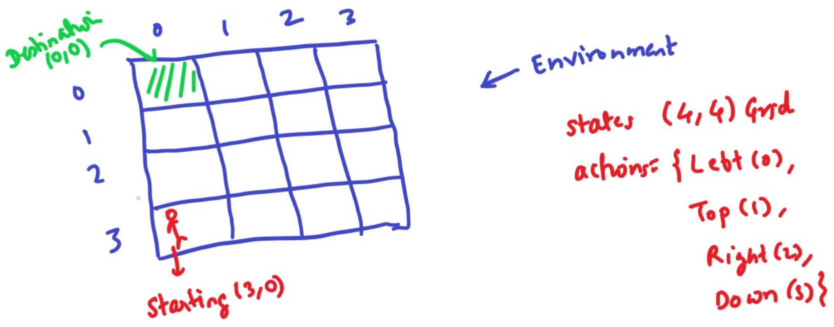
First we discuss an example of Markov decision processes for your understanding. We will have questions based on this in subsequent sections. Link of working code is provided here.

1.1 Environment

1. Here, the environment is a fixed MDP (Markov Decision Process) environment, with a fixed number of states, and a fixed number of actions in each of the states.
2. The environment is basically a $(4, 4)$ grid, in which the top left corner is indexed as $(0, 0)$ and the bottom right corner is indexed as $(3, 3)$.
3. The dynamics of the environment, i.e., $P(s'|s, a)$ is pretty straight-forward. For a given (state, action) pair, there exists only a single (next-state), and hence the dynamics function is either 0 or 1. This is referred to as deterministic model of environment.

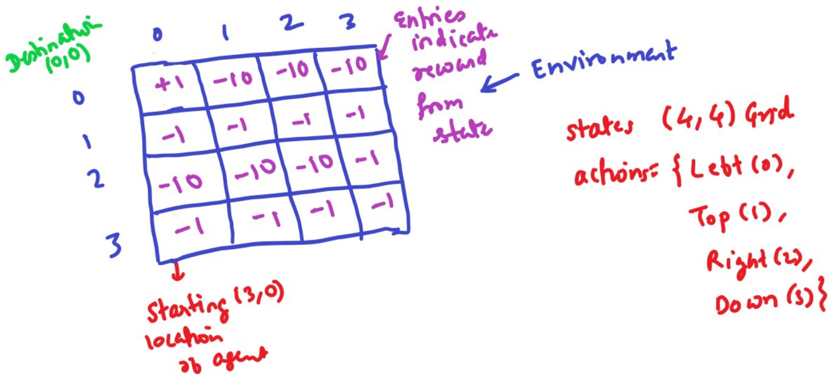
1.2 States and actions

1. Each state is represented by a list of 2 indices, the first index denotes the row and the second index denotes the column.
2. The agent starts from $(3, 0)$, i.e., bottom left corner and has to reach $(0, 0)$, i.e., top left corner, which is the terminal state.
3. In any state, the agent has 4 possible actions, Left (0), Top (1), Right (2) and Down (3).
4. Any action that is supposed to make the agent leave the grid, will bring back the agent in the same state, along with the reward for that state.



1.3 Rewards

1. The distribution of rewards is stationary and the reward in each state is also a single number.
2. The below image depicts the rewards that the agent receives in each of the states.
3. This is the default rewards distribution feeded into the environment, however, if we want, we can change this distribution as well, by passing in a custom reward distribution too.



	0	1	2	3
0	+1	-10	-10	-10
1	-1	-1	-1	-1
2	-10	-10	-10	-1
3	-1	-1	-1	-1

1.4 Policy iteration algorithm

The environment begins with an equiprobable random policy or fixed policy and an all-zero state value function. The task of the reinforcement learning algorithm is to find out the optimal state value function and the optimal policy.

Policy iteration algorithm: It involves two steps—1) policy evaluation and 2) policy improvement

1.4.1 Policy evaluation

For given policy π and starting state s, the expected discounted cumulative reward for infinite horizon is as follows.

$$\begin{aligned} V^\pi(s) &= \mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\beta^t r_t \;\middle|\; S_0 = s\right] && (1)\\ &= r(s,a) + \beta\mathbb{E}^\pi\left[\sum_{t=1}^{\infty}\beta^(t-1) r_t \;\middle|\; S_1 = s'\right] && (2)\\ &= r(s,a) + \beta\sum_{s'\in\mathcal{S}} p^a_{s,s'} V^\pi(s') && (3)\end{aligned}$$

where $a = \pi(s)$ action under policy π for given state s.

$V^\pi(s)$ is called as state value function under policy π.

We want to compute $V^\pi(s)$ for given s and policy π. The computation algorithm for $V^\pi(s)$ is policy evaluation or prediction problem. It is an iterative algorithm. One start with initial value for each state $s \in \mathcal{S}$, and update state value function using following iterative scheme.

$$V^\pi_{t+1}(s) = r(s,a) + \beta\sum_{s'\in\mathcal{S}} p^a_{s,s'} V^\pi_t(s'), \qquad (4)$$

where $a = \pi(s)$, this is computed for all s. This can be run until $t = 1000$ or $\Delta_{t+1} < \theta$ where $\Delta_{t+1} = \max\{\Delta_t, \max_s|V^\pi_{t+1}(s) - V^\pi_t(s)|\}$.

Reading exercise: Chapter 4 From Introduction to Reinforcement learning, Dynamic Programming

1.4.2 Policy improvement

The action value function for policy π is given by

$$\begin{aligned} q^\pi(s,a) &= \mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\beta^t r_t \;\middle|\; S_0 = s, A_0 = a\right] && (5)\\ &= r(s,a) + \beta\mathbb{E}^\pi\left[\sum_{t=1}^{\infty}\beta^(t-1) r_t \;\middle|\; S_1 = s', A_1 = \pi(s')\right] && (6)\\ &= r(s,a) + \beta\sum_{s'\in\mathcal{S}} p^a_{s,s'} V^\pi(s') && (7)\end{aligned}$$

Here, choose action a in first step and in later steps follow policy π.

$q^\pi(s,a)$ is called as state-action value function under policy π.

Our reason for computing the value function for a policy is to help find better policies. Suppose we have determined the value function V^π for an arbitrary deterministic policy π. For some state s we would like to know whether or not we should change the policy to deterministically choose an action $a \neq \pi(s)$. We know how good it is to follow the current policy from s—that is $V^\pi(s)$—but would it be better or worse to change to the new policy? One way to answer this question is to consider selecting a in s and thereafter following the existing policy, π. This gives us.

$$q^\pi(s,a) = r(s,a) + \beta\sum_{s'\in\mathcal{S}} p^a_{s,s'} V^\pi(s')$$

We consider new greedy policy $\pi'(s)$ as follows.

$$\begin{aligned} \pi'(s) &= \arg\max_a q^\pi(s,a) && (8)\\ &= \arg\max_a\left\{r(s,a) + \beta\sum_{s'\in\mathcal{S}} p^a_{s,s'} V^\pi(s')\right\} && (9)\end{aligned}$$

for all $s \in \mathcal{S}$.

This is referred to as *policy improvement* step. Here

$$V^{\pi'}(s) = \max_a \left\{ r(s,a) + \beta \sum_{s' \in \mathcal{S}} p^a_{s,s'} V^{\pi}(s') \right\} \tag{10}$$

$$= \max_a q^{\pi}(s,a) \tag{11}$$

If policy π' is better than policy π for all $s \in \mathcal{S}$ then it implies that

$$V^{\pi'}(s) \geq V^{\pi}(s). \tag{12}$$

The policy improvement step is discussed in greater detail in Section 4.3, Chapter 4, Introduction to Reinforcement Learning by Sutton and Barto.

1.5 Value iteration algorithm

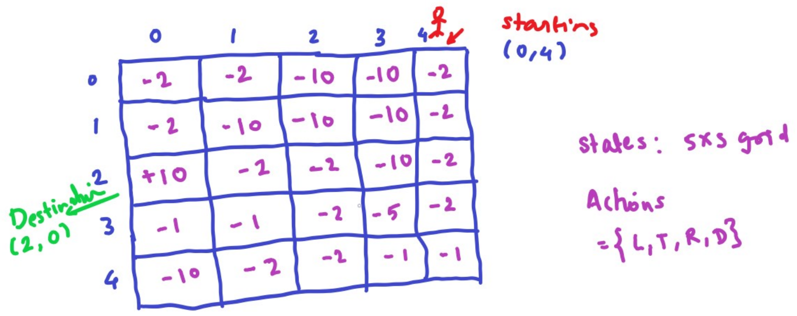
One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to V^{π} occurs only in the limit. Must we wait for exact convergence, or can we stop short of that? The value iteration can be written as a particularly simple update operation that combines the policy improvement and truncated policy evaluation steps:

$$V_{t+1}(s) = \max_a \left\{ r(s,a) + \beta \sum_{s' \in \mathcal{S}} p^a_{s,s'} V_t(s') \right\}, \tag{13}$$

for all $s \in \mathcal{S}$. Starting from arbitrary V_0, the sequence $\{V_t\}$ converges to V^* the optimal state value function. This is also referred to as Bellman optimality condition. Detail on value iteration algorithm is discussed in Section 4.4, Chapter 4, Introduction to reinforcement learning by Sutton and Barto.

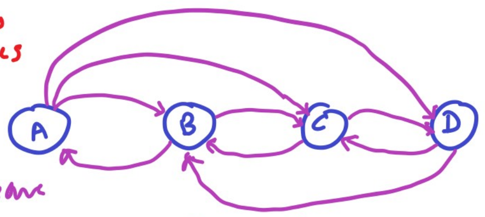
Questions 1: Basic code on policy iteration and value iteration is shared with you for one example. You must go through code and see the effect of discount parameter β, rewards by changing it structure. 1) Compute the policy for $\beta = 0.5, 0.9, 0.99$ Show its plot. 2) Change reward structure at location $(1,0)$ to -3 and -5 and show plots for policy iteration and value iteration.

Questions 2: Basic code on policy iteration and value iteration is shared with you for one example. Modify the basic code for grid size of $(5,5)$ as given in Fig. and obtain plots for policy iteration and value iteration.



Questions 3: Consider an environment that consists of 4 states $\{A, B, C, D\}$ and an agent with 2 actions $\{S, L\}$. Action L corresponds to leave and action S corresponds to stay/go back. Each state could represent

Transitions
for '4' states
'2' actions

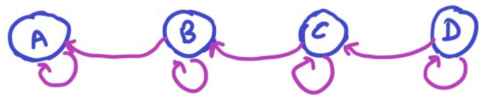


Action: Leave

(for leave action)

Transition prob mx
$$P^L = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 0.2 & 0.3 & 0.5 \\ B & 0.5 & 0 & 0.5 & 0 \\ C & 0 & 0.5 & 0 & 0.5 \\ D & 0 & 0.5 & 0.5 & 0 \end{array}$$

Transitions
for '4' states
'2' actions



action: S (stay/Goback)

(for action S: stay/Go back)

Transition prob mx
$$P^S = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 1 & 0 & 0 & 0 \\ B & 0.2 & 0.8 & 0 & 0 \\ C & 0 & 0.2 & 0.8 & 0 \\ D & 0 & 0 & 0.2 & 0.8 \end{array}$$

<u>Rewards:</u>

$$R = \begin{array}{c|cc} & L & S \\ \hline A & 2 & 3 \\ B & 1 & 2 \\ C & 3 & 1 \\ D & 10 & 0 \end{array} \leftarrow \text{actions}$$

$\uparrow$ states

the location of agent where he want to be. Each action from the given state yields reward that depends on action of agent. The transition probability matrices and rewards are given in Figures.

Suppose that agent follows a policy $\pi = \{S, S, S, S\}$ for each state, what is the state value function $V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \beta^t r_t \;\middle|\; S_0 = s\right]$ for $s \in \{A, B, C, D\}$.

Write a program in python for this model and compute the $V^{\pi}(s)$. Also implement the policy iteration and value iteration algorithm.

2 Part B: Q-learning algorithm

Suppose that transition probability matrix and reward matrix are unknown. But agent observes the data $\{s_t = s, a_t = a, r_{t+1} = r, s_{t+1} = s'\}$. Q-learning algorithm is model free algorithm, where we learn $Q(s, a)$ state-action value function directly from data in sequentially.

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha_{t+1}\left[r_{t+1} + \beta \max_{a'} Q_t(s', a') - Q_t(s, a)\right] \tag{14}$$

An action selection happens according to ϵ-greedy policy.

Question 1 Write a Q-learning program for example discussed in Question 3 in previous section. Show the evolution of Q_t as function of t (iteration), for discount parameter $\beta = 0.8$, $\epsilon = 0.3$. Choose appropriate stepsizes.

Does this converges to optimal Q^* optimal state-action value function? If yes, provide justification. When this possible?

Question 2 I have shared code with Q-learning code and value iteration. Understand the code given to you give a plot for discount parameters $\beta = 0.9$, and $\beta = 0.5$