Markov Decision Problem and Reinforcement Learning Exercises

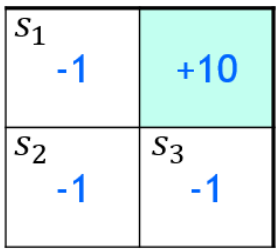
Prof. Jim Rehg
CS 3600 Introduction to Artificial Intelligence
School of Interactive Computing
Georgia Institute of Technology

February 28, 2018

These practice exercises are for your benefit in preparing for the exams. They will not be collected or graded. Solutions will be provided. Note that not all questions are representative of questions you will find on the exam, but the material covered by these questions will also be covered by the exams.

Question 1. **Markov Decision Problem**

Consider the simple grid-world MDP shown below. The agent starts at random in one of three states S_1 to S_3 and must reach the shaded terminal square and collect the reward of +10. All nonterminal states have a reward of -1. Assume no discounting ($\gamma = 1$). The available actions are *Left*, *Right*, *Up*, and *Down*. Use the standard probabilistic transition model for actions in which the probability of moving in the desired direction is 0.8 and there is 0.1 chance to move in each of the orthogonal directions (i.e. there is no probability of moving backwards).



(a) (6 pts) Assuming that the utilities are initialized to zero, perform *two rounds of value iteration*, showing the updated utilities after each round. In order to speeed convergence, use the most recent utility values that are available (i.e. use utilities computed earlier in the same iteration whenever they are available.)

SOLUTION:

From the structure of the problem, it will never be advantageous for the agent to drive into the walls, therefore we only need to consider the actions in each state which move to an adjacent state or the terminal goal (i.e. the third action choice will never be selected). It is clear that there is a natural symmetry in the problem, and we will have $U(S_1) = U(S_3)$ under the optimal policy (and any other symmetric policy). Using the most recent values during value iteration will break this symmetry temporarily, but the final value function will have it.

$$U^1(S_1) = -1 + \max_{R,D} \begin{cases} R : 0.8 \times 10 + 0 = \underline{8} \\ D : 0 + 0.1 \times 10 = 1 \end{cases} \quad = 7$$

$$U^1(S_2) = -1 + \max_{R,U} \begin{cases} R : 0.1 \times 7 = 0.7 \\ U : 0.8 \times 7 + 0 = \underline{5.6} \end{cases} \quad = 4.6$$

$$U^1(S_3) = -1 + \max_{U,L} \begin{cases} U : 0.8 \times 10 + 0.1 \times 4.6 = \underline{8.46} \\ L : 0.8 \times 4.6 + 0.1 \times 10 = 4.68 \end{cases} \quad = 7.46$$

$$U^2(S_1) = -1 + \max_{R,D} \begin{cases} R : 0.8 \times 10 + 0.1 \times 4.6 + 0.1 \times 7 = \underline{9.16} \\ D : 0.8 \times 4.6 + 0.1 \times 10 + 0.1 \times 7 = 5.38 \end{cases} = 8.16$$

$$U^2(S_2) = -1 + \max_{R,U} \begin{cases} R : 0.8 \times 7.46 + 0.1 \times 8.16 + 0.1 \times 4.6 = 7.24 \\ U : 0.8 \times 8.16 + 0.1 \times 7.46 + 0.1 \times 4.6 = \underline{7.73} \end{cases} = 6.73$$

$$U^2(S_3) = -1 + \max_{U,L} \begin{cases} U : 0.8 \times 10 + 0.1 \times 6.73 + 0.1 \times 7.46 = \underline{9.42} \\ L : 0.8 \times 6.73 + 0.1 \times 10 + 0.1 \times 7.46 = 7.13 \end{cases} = 8.42$$

Note that if you recognized the symmetry in the problem and *chose not to compute updates* for $U^i(S_3)$ that is completely fine and you would receive full credit, as long as you stated your reasons clearly. Note also that answer is not affected by the randomized starting state.

(b) (3 pts) Use *policy iteration* to solve for the optimal value function in one step, by identifying the optimal policy and then using policy evaluation. *Hint:* Solve the linear system of equations given by the Bellman equations. Verify that these are indeed the optimal utilities by showing that they cause *value iteration* to terminate.

SOLUTION:

It's clear that the optimal policy is $\pi(S_1) = R, \pi(S_2) = U$ or $R, \pi(S_3) = U$. We can choose $\pi(S_2) = R$ arbitrarily. The Bellman equations for a fixed policy are:

$$V(S_t) = R(S_t) + \gamma \sum_{S_{t+1}} P(s_{t+1}|s_t, \pi(s_t))V(s_{t+1}).$$

This gives an equation for each state:

$$\begin{aligned} V(S_1) = &\quad -1 + (0.8)10 + 0.1V(S_2) + 0.1V(S_1) \\ V(S_2) = &\quad -1 + 0.8V(S_3) + 0.1V(S_1) + 0.1V(S_2) \\ V(S_3) = &\quad -1 + (0.8)10 + 0.1V(S_2) + 0.1V(S_3) \end{aligned}$$

Defining $x_i = V(S_i)$ and rearranging gives:

$$\begin{array}{rcrcrcl} 0.9x_1 & - & 0.1x_2 & & & = & 7 \\ -0.1x_1 & + & 0.9x_2 & - & 0.8x_3 & = & -1 \\ & - & 0.1x_2 & + & 0.9x_3 & = & 7 \end{array}$$

Eliminating x_1 yields: $8x_2 - 7.2x_3 = -2$. Eliminating x_2 yields $x_3 = 8.611$. Substituting into the last equation yields $x_2 = 7.5$. It follows that:

$$V(S_1) = V(S_3) = 8.611 \qquad V(S_2) = 7.5$$

We now show that the values (and therefore the policy) are optimal by demonstrating that they are a stationary point for value iteration:

$$V(S_1) = -1 + \max_{R,D} \begin{cases} R : 0.8 \times 10 + 0.1 \times 7.5 + 0.1 \times 8.611 = \underline{9.611} \\ D : 0.8 \times 7.5 + 0.1 \times 10 + 0.1 \times 8.611 = 7.86 \end{cases} = 8.611$$

$$V(S_2) = -1 + \max_{R,U} \begin{cases} R : 0.8 \times 8.611 + 0.1 \times 8.611 + 0.1 \times 7.5 = 8.5 \\ U : 0.8 \times 8.611 + 0.1 \times 8.611 + 0.1 \times 7.5 = 8.5 \end{cases} = 7.5$$

$V(S_3) = V(S_1) = 8.611$ (by symmetry)

It follows that subsequent iterations will not change the values, therefore they are optimal and so is the policy.

(c) (2 pts) Compute the table $Q^*(s, a)$ for the optimal policy π^* from part (b). The table is indexed by (s, a) pairs, where $Q^*(s, a)$ is the expected discounted reward obtained by starting in state s, collecting $R(s)$, taking action a, and subsequently following π^*.

SOLUTION:

Although the table is 3 by 4, in every state there are two actions we will never take, so we don't need to compute their Q values. We first note that $Q(S_1, R) = V(S_1)$ and $Q(S_3, U) = V(S_3)$ because these are the optimal actions and therefore the expected discounted reward for taking those actions is just the utility of the state. Likewise, we have $Q(S_2, U) = Q(S_2, R) = V(S_2)$ because both actions are optimal due to symmetry. There is only one table entry that is missing: $Q(S_1, D)$, because also by symmetry we have $Q(S_1, D) = Q(S_3, L)$. By definition:

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a)V(s').$$

Substituting from the MDP and value function gives:

$$Q(S_1, D) = Q(S_3, L) = -1 + 0.8 \times 7.5 + 0.1 \times 1 + 0.1 \times 8.611 = 6.8611$$

(d) (1 pts) You have obtained the following episodes from running your agent through the grid world, and would like to use Q-Learning.

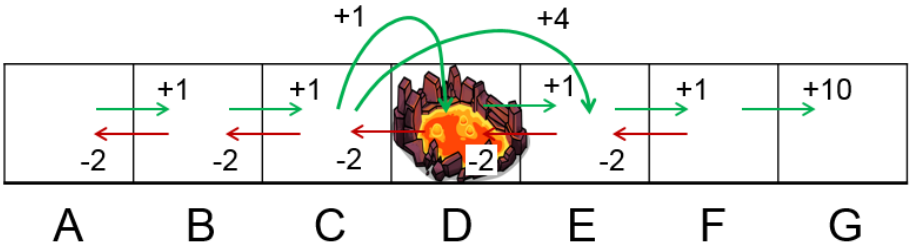
Episode 1	Episode 2	Episode 3
$S_1, D, S_2, -1$	$S_2, R, S_3, -1$	$S_3, L, S_2, -1$
$S_2, R, S_3, -1$	$S_3, L, S_2, -1$	$S_2, U, S_1, -1$
$S_3, U, T, +10$	$S_2, R, S_3, -1$	$S_1, R, T, +10$
	$S_3, U, T, +10$	

Will Q-Learning be able to learn $Q^*(s, a)$ from part (c) based on these episodes? Why or why not?

NO. The episode data does not contain any examples of the probabilistic transition model. Every action resulted in a correct transition. As a consequence, the Q-Learner will not be able to incorporate the probabilistic transition model into the Q function. For example, based on the data $Q(S_3, U) = 10$. Note that the agent in these episodes is clearly not following the optimal policy. This is not a concern in principle, since Q-Learning is an off-policy method.

Question 2. American Ninja MDP

You need to solve the following MDP to prepare for your upcoming appearance on the hit TV show *American Ninja Warrior*. Your objective is to start at state A, race down the track, jump the lava pit in state D, and reach the finish line (terminal state G). Actions are *Right*, *Left*, and *Jump*, but *Jump* can only be used in state C (and *Right* cannot be used there). Rewards for each state transition are shown in the figure. The discount is $\gamma = 1$.



All actions are *deterministic* except for *Jump*, which succeeds half the time, landing in state E, and fails half the time, landing in D. In summary, the action model is:

Right: Deterministically move to the right.
Left: Deterministically move to the left.
Jump: Stochastically jump to the right. This action is available for square C only.
$T(C, \textit{Jump}, E) = 0.5$ (jump succeeds)
$T(C, \textit{Jump}, D) = 0.5$ (jump fails)

(a) (2 pts) For the policy π of always moving forward (i.e., using actions *Right* or *Jump*), compute $V^{\pi}(C)$.

Since there is no discounting, the value is the sum of the rewards. If the jump fails, reward to go from C is $1 + 1 + 1 + 10 = 13$. If the jump succeeds it is $4 + 1 + 10 = 15$. Since both are equally likely, $V^{\pi}(C) = (13 + 15)/2 = 14$.

(b) (3 pts) Perform two iterations of value iteration and fill in the table below. All values are initialized to zero.

$V^2(B)$	3.5
$Q^2(B, \textit{Right})$	3.5
$Q^2(B, \textit{Left})$	-1

SOLUTION:

The update rule for reward on transition is:

$$V^{i+1}(s) = \max_a \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V^i(s')\right]$$

For state C this gives:

$$V^{i+1}(C) = \max_{J,L} \begin{cases} J : 0.5(1 + V^i(D)) + 0.5(4 + V^i(E)) \\ L : -2 + V^i(B) \end{cases}$$

For the other states, each possible action gives its reward plus the value of the successor state. The following table shows the values for each iteration.

	A	B	C	D	E	F	G
V^1	1	1	2.5	1	1	10	0
V^2	2	3.5	3.5	2	11	10	0

The expression for the Q values is:

$$Q(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right],$$

which implies that $V^i(s) = \max_a Q^i(s,a)$. For state B:

$$V^2(B) = \max_{R,L} \begin{cases} R: 1 + 2.5 = Q^2(B,R) = 3.5 \\ L: -2 + 1 = Q^2(B,L) = -1 \end{cases} = 3.5.$$

(c) (3 pts) You decide to use Q-Learning to obtain the optimal policy. After some number of iterations of Q-Learning, the Q table has the values given in the row "Initial" below. Apply the Q-Learning update rule to update the Q-values according to the four transitions in the episode given below. Q-values that are unaffected by a particular transition can be left blank. Use a learning rate α of 0.5. *Be sure to use the initial Q-values provided in the top row.*

Episode Data for Q-Learning

s	a	r	s	a	r	s	a	r	s	a	r	s
C	*Jump*	$+4$	E	*Right*	$+1$	F	*Left*	-2	E	*Right*	$+1$	F

	$Q(\text{C}, \textit{Left})$	$Q(\text{C}, \textit{Jump})$	$Q(\text{E}, \textit{Left})$	$Q(\text{E}, \textit{Right})$	$Q(\text{F}, \textit{Left})$	$Q(\text{F}, \textit{Right})$
Initial	-1	1	0	2	0	-2
Transition 1		3.5				
Transition 2				1.5		
Transition 3					-0.25	
Transition 4				1.125		

SOLUTION:

The Q-learning update rule for transition i is

$$Q^i(s,a) = (1-\alpha)Q^{i-1}(s,a) + \alpha\left[R(s,a,s') + \max_{a'}\gamma Q^{i-1}(s',a')\right].$$

Details of the calculations:

$$\begin{aligned} 3.5 &= 0.5(1) + 0.5(4+2) \\ 1.5 &= 0.5(2) + 0.5(1+0) \\ -0.25 &= 0 + 0.5(-2+1.5) \\ 1.125 &= 0.5(1.5) + 0.5(1-0.25) \end{aligned}$$

A state-action is only updated when a transition is made from it. $Q(C, \textit{Left}), Q(E, \textit{Left}),$ and $Q(F, \textit{Right})$ state-actions are never experienced and so these values are never updated.