Probabilistic Artificial Intelligence
Solutions to Problem Set 5

Nov 25, 2019

1. Value iteration

In finite MDPs, the value function can be expressed as a vector that has as many entries as states in the state space, X. Given a value vector V, we defined the Bellman update operator, $\mathcal{B}(\cdot)$, for every element of V as follows:

$$\mathcal{B}(V(x)) = \max_a \big(r(x,a) + \gamma \sum_{x'} P(x'|x,a)V(x')\big). \tag{1}$$

Show that the Bellman operator is a contraction with respect to $\|\cdot\|_\infty$, that is to say that, for any V, V', holds that:

$$\max_{x\in X} |\mathcal{B}(V(x)) - \mathcal{B}(V'(x))| = \|\mathcal{B}V - \mathcal{B}V'\|_\infty \leq \gamma \|V - V'\|_\infty. \tag{2}$$

Solution

By considering a generic $x \in X$ we can write:

$$\begin{aligned}
&|\max_a \big(r(x,a) + \gamma \sum_{x'} P(x'|x,a)V(x')\big) - \max_a \big(r(x,a) + \gamma \sum_{x'} P(x'|x,a)V'(x')\big)| \\
&\leq \max_a \gamma \left| \sum_{x'} P(x'|x,a)V(x') - \sum_{x'} P(x'|x,a)V'(x') \right| \\
&= \max_a \gamma \left| \sum_{x'} P(x'|x,a)(V(x') - V'(x')) \right| \\
&\leq \max_a \gamma \sum_{x'} P(x'|x,a) \left|V(x') - V'(x')\right| \\
&\leq \gamma \max_{x'} |V(x') - V'(x')| \\
&= \gamma \|V - V'\|_\infty.
\end{aligned}$$

We proved that this inequality holds for a generic $x \in X$. This means that it must hold in particular for the value that attains the maximum of the left hand side of the inequality:

$$|\|\mathcal{B}V - \mathcal{B}V'\|_\infty = \max_{x\in X} |\mathcal{B}(V(x)) - \mathcal{B}(V'(x))| \leq \gamma \|V - V'\|_\infty. \tag{3}$$

2. Push Your Luck

Consider the following game played over multiple rounds. In each round, you have the option to either push your luck (P) or leave (L). If you leave, you obtain 0 reward and the game ends.

Otherwise, you draw a card uniformly at random from a deck of cards. In the deck of cards, you have lucky and unlucky cards. When a lucky card is drawn, you obtain a positive reward shown on the card, the card is removed from the deck, and you continue to the next round. When an unlucky card is drawn, you obtain a negative reward shown on the card, and the game ends immediately.

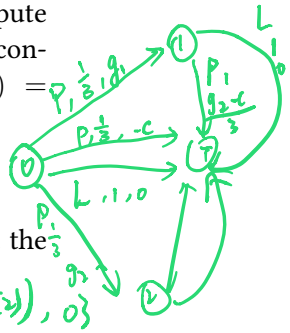
For all the following questions, we consider only two rounds of the game, that is, the game ends after (at most) two choices of P or L. Furthermore, the deck consists of three cards, two lucky and one unlucky card, with rewards g_1, g_2, and $-c$, respectively, where $g_1, g_2, c > 0$.

(i) Draw an MDP diagram that encodes this game, annotating the transitions with the corresponding actions, transition probabilities, and associated rewards.

(ii) Assume that $g_1 > c$ and $g_2 < c$. For the undiscounted version ($\gamma = 1$), compute the optimal values of all (non-terminal) states by running value iteration until convergence. Initialize all values to 0. *(Hint: The Value Iteration update is* $V_{k+1}(x) = \max_a \{\sum_{x'} P(x' \mid x, a)(r(x, a, x') + V_k(x'))\}$*).*

(iii) Compute the optimal policy using the result of the previous question.

(iv) If $g_1 < c$ and $g_2 < c$, but $g_1 + g_2 > c$, briefly show how the optimal values and the optimal policy computed in the previous questions change.



Solution

1. See Figure 1.

2. Iteration 1:

$$
\begin{aligned}
V_1(0) &= \max\{1/3(g_1 + V_0(1)) + 1/3(g_2 + V_0(2)) + 1/3(-c + 0), 0\} \\
&= 1/3(g_1 + g_2 - c) \\
V_1(1) &= \max\{1/2(g_2 + 0) + 1/2(-c + 0), 0\} \\
&= 0 \\
V_1(2) &= \max\{1/2(g_1 + 0) + 1/2(-c + 0), 0\} \\
&= 1/2(g_1 - c)
\end{aligned}
$$

Iteration 2:

$$
\begin{aligned}
V_2(0) &= \max\{1/3(g_1 + V_1(1)) + 1/3(g_2 + V_1(2)) + 1/3(-c + 0), 0\} \\
&= 1/3(g_1 + g_2 + 1/2(g_1 - c) - c) \\
&= 1/2(g_1 - c) + 1/3 g_2 \\
V_2(1) &= \max\{1/2(g_2 + 0) + 1/2(-c + 0), 0\} \\
&= 0 \\
V_2(2) &= \max\{1/2(g_1 + 0) + 1/2(-c + 0), 0\} \\
&= 1/2(g_1 - c)
\end{aligned}
$$

Already converged ($V_3(0)$) depends only on $V_2(1)$ and $V_2(2)$ which already converged.

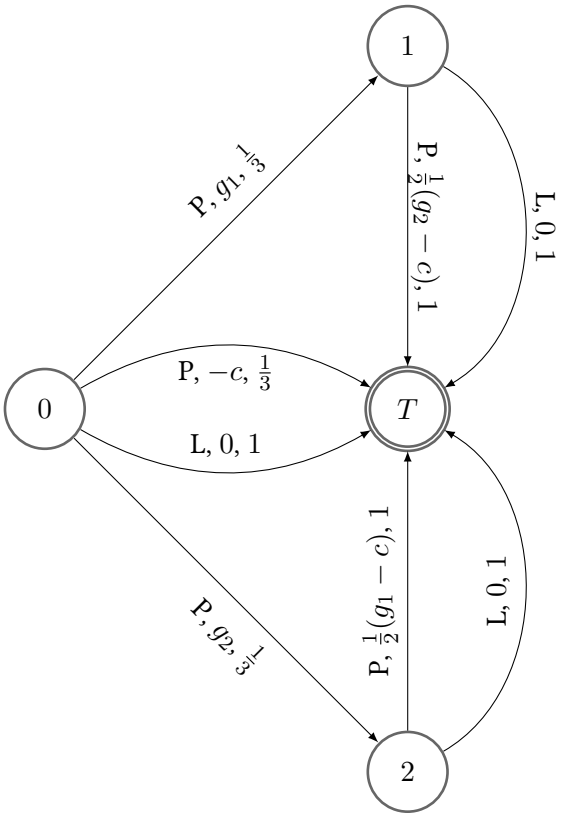


Figure 1: MDP diagram encoding the game of "Push Your Luck". Each state is a node of the graph. Each edge with a triple A, R, p encodes a state transiton when chosing action A, the reward is R, and the transition probability is p.

3.

$$\begin{aligned}
\pi^\star(0) &= \arg\max\{1/2(g_1 - c) + 1/3g_2, 0\} \\
&= \text{P} \\
\pi^\star(1) &= \arg\max\{1/2(g_2 - c), 0\} \\
&= \text{L} \\
\pi^\star(2) &= \arg\max\{1/2(g_1 - c), 0\} \\
&= \text{P}
\end{aligned}$$

4. We have that $V^\star(1) = V^\star(2) = 0$, and $V^\star(0) = 1/3(g_1 + g_2 - c)$.
   Equivalent, the policy is $\pi^\star(1) = \pi^\star(2) = \text{L}$, and $\pi^\star(0) = \text{P}$.

3. Server Scheduling

Consider a server that serves two queues indexed by $i = 1, 2$. Queue i receives a new request at time-steps $2t + i$ ($t \in \mathbb{Z}$). Each request remains in a queue for the one time step after it has

been received and expires afterwards, so that each queue can contain at most one request at a time.

Let q_t specify the queue the server serves at time t. Serving a queue proceeds as follows: if the queue is non-empty, then the server clears the request from that queue and earns a reward of 1; otherwise the server receives no reward.

At the end of each time step t, the server decides whether to remain at the current queue (i.e., $q_{t+1} = q_t$) or move to the other queue (i.e., $q_{t+1} \neq q_t$). If the server chooses to move, it incurs an additional cost of $C \geq 0$; otherwise, no cost is incured.

(i) Draw the MDP described above, annotating the action-dependent transitions with transition probabilities and associated rewards.

(ii) Let the discount factor be $\gamma = 0.5$ and the cost be $0 \leq C < 1/3$. Guess what the optimal policy is and solve the Bellman equations to compute the value function for that policy.

(iii) Using the result of the previous question, show that the policy you guessed is indeed optimal.

Solution

1. The MDP consists of 4 states, specifying the current location of the server ($q_t \in \{1, 2\}$), and the parity of the current time (t is odd or even).

   - At state $(q_t = 1, t = e)$, action "stay" takes us deterministically to $(q_t = 1, t = o)$ and action "move" takes us deterministically to $(q_t = 2, t = o)$. Rewards are $0, -C$, respectively.
   - At state $(q_t = 1, t = o)$, action "stay" takes us deterministically to $(q_t = 1, t = e)$ and action "move" takes us deterministically to $(q_t = 2, t = e)$. Rewards are $1, 1 - C$, respectively.
   - At state $(q_t = 2, t = e)$, action "stay" takes us deterministically to $(q_t = 2, t = o)$ and action "move" takes us deterministically to $(q_t = 1, t = o)$. Rewards are $1, (1 - C)$, respectively.
   - At state $(q_t = 2, t = o)$, action "stay" takes us deterministically to $(q_t = 2, t = e)$ and action "move" takes us deterministically to $(q_t = 1, t = e)$. Rewards are $0, -C$, respectively.

2. We guess that the following policy π is optimal:

   - At $s_1 = (q_t = 1, t = e)$, stay.
   - At $s_2 = (q_t = 2, t = o)$, stay.
   - At $s_3 = (q_t = 1, t = o)$, move.
   - At $s_4 = (q_t = 2, t = e)$, move.

The state values associated with π can be calculated as follows:

$$\begin{aligned} v_1 &= 0 + \gamma v_3 \\ v_2 &= 0 + \gamma v_4 \\ v_3 &= (1 - C) + \gamma v_4 \\ v_4 &= (1 - C) + \gamma v_3 \end{aligned}$$

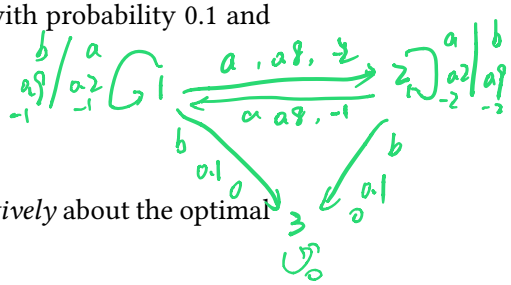
which results in $v_1 = v_2 = \frac{\gamma(1-C)}{1-\gamma}$, and $v_3 = v_4 = \frac{(1-C)}{1-\gamma}$.

3. Since $0 < \gamma < 1$ and $0 \leq C < 1$, we have that $\frac{\gamma(1-C)}{1-\gamma} < \frac{(1-C)}{1-\gamma}$, so it is easy to see that π is the optimal policy with respect to V_π.

4. Policy iteration

Consider an undiscounted MDP having three states, (1, 2, 3), with rewards -1, -2, 0, respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: a and b. The transition model is as follows:

- In state 1, action a moves the agent to state 2 with probability 0.8 and makes the agent stay put with probability 0.2.
- In state 2, action a moves the agent to state 1 with probability 0.8 and makes the agent stay put with probability 0.2.
- In either state 1 or state 2, action b moves the agent to state 3 with probability 0.1 and makes the agent stay put with probability 0.9.



Answer the following questions:

(i) Draw the MDP described above. What can be determined *qualitatively* about the optimal policy in states 1 and 2?

(ii) Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action b in both states.

(iii) What happens to policy iteration if the initial policy has action a in both states? Does discounting help? Does the optimal policy depend on the discount factor?

Solution

(i) Intuitively, the agent wants to get to state 3 as soon as possible, because it will pay a cost for each time step it spends in state 1 and state 2. However, the only action that reaches state 3 (action b) succeeds with low probability, so the agent should minimize the cost it incurs while trying to reach the terminal state. This suggests that the agent should definitely try action b in state 1; in state 2, it might be better to try action a to get to state

1 (which is the better place to wait for admission to state 3), rather than aiming directly for state 3. The decision in state 2 involves a numerical tradeoff.

(ii) The application of policy iteration precedes in alternating steps of value determination and policy update.

- Initialization: $U \leftarrow \langle -1, -2, 0 \rangle$, $P \leftarrow \langle b, b \rangle$.
- Value determination: Write out the equations in terms of the values (rewards and transition probabilities are known for a fixed policy $\pi(x)$)

$$u(x) = r(x) + \sum_{x'} P(x'|x, \pi(x))u(x')$$

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.1u_3 + 0.9u_2 \\ u_3 &= 0 \end{aligned}$$

Which have the solution, $u_1 = -10$ and $u_2 = -20$.
Policy update:
The reward is not dependent on the action, so it does not affect the maximization problem and is neglected in the following. In state 1,

$$\sum_j T(1, a, j)u_j = 0.8 \times -20 + 0.2 \times -10 = -18$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 + 0.9 \times -10 = -9$$

so action b is preferred for state 1.
In state 2,

$$\sum_j T(1, a, j)u_j = 0.8 \times -10 + 0.2 \times -20 = -12$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 + 0.9 \times -20 = -18$$

so action a is preferred for state 2. We set $unchanged? \leftarrow$ false and proceed.

- Value determination:

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

once more, $u_1 = -10$; now, $u_2 = -12.5$.
Policy update:
In state 1,

$$\sum_j T(1, a, j)u_j = 0.8 \times -12.5 + 0.2 \times -10 = -12$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 + 0.9 \times -10 = -9$$

so action b is still preferred for state 1.
In state 2,

$$\sum_j T(1, a, j)u_j = 0.8 \times -10 + 0.2 \times -12.5 = -10.5$$

while

$$\sum_j T(1, b, j)u_j = 0.1 \times 0 + 0.9 \times -12.5 = -11.25$$

so action a is still preferred for state 2. $unchanged?$ remains true, and we terminate.

Note that the resulting policy matches our intuition: when in state 2, try to move to state 1, and when in state 1, try to move to state 3.

(iii) An initial policy with action a in both states leads to an unsolvable problem. The initial value determination problem has the form

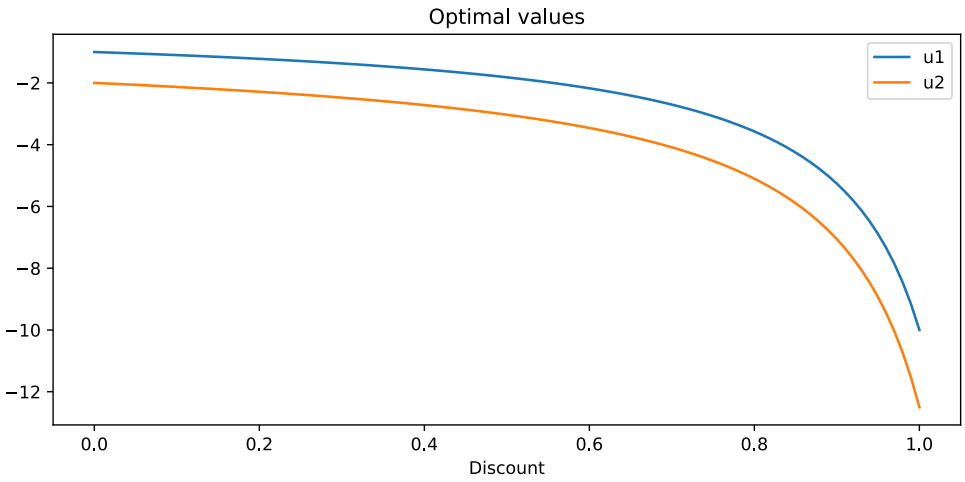
$$\begin{aligned} u_1 &= -1 + 0.2u_1 + 0.8u_2 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

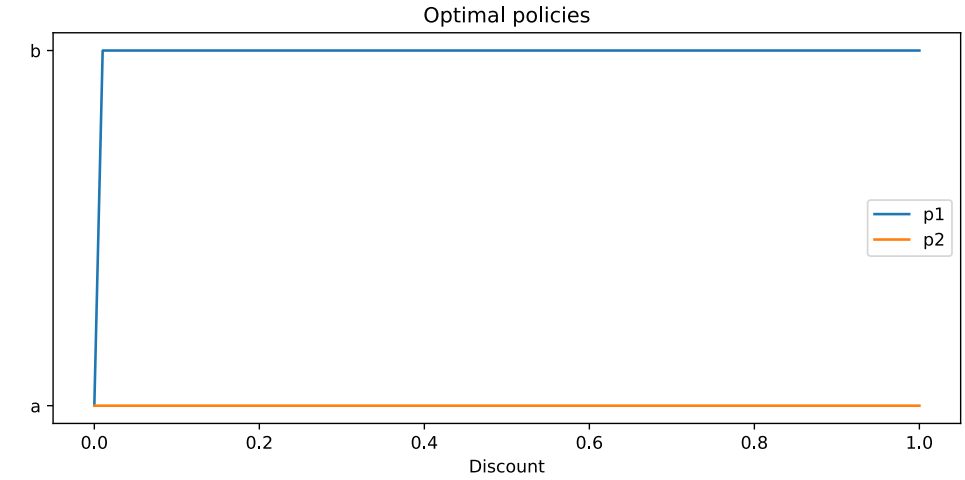
and the first two equations are inconsistent. If we were to try to solve them iteratively, we would find the values tending to $-\infty$.

Discounting leads to well-defined solutions by bounding the penalty (expected discounted cost) an agent can incur at either state. For this problem, the discount factor γ does not affect the optimal policy, as shown in the figure below. (For $\gamma = 0$, all policies are optimal, since future rewards are completely disregarded.)

$0.8u_1 = -1 + 0.8u_2$

$0.8u_2 = -2 + 0.8u_1$


Optimal values
u1
u2
−2
−4
−6
−8
−10
−12
0.0
0.2
0.4
0.6
0.8
1.0
Discount


Optimal policies
b
a
p1
p2
0.0
0.2
0.4
0.6
0.8
1.0
Discount