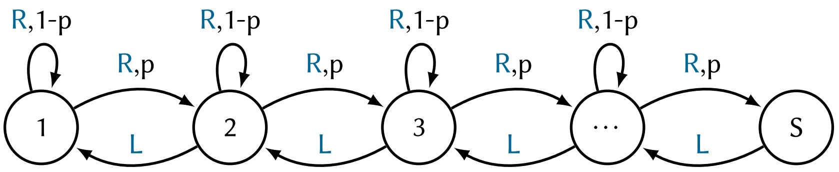
Probabilistic Artificial Intelligence
Problem Set 6

Dec 14, 2018

1. River Swim

Consider the MDP with states $1, \ldots, S$ as illustrated below. The agent starts in state 1 and tries to swim up the river to reach the final state S. All states except for 1 and S have two actions left (L) and right (R). The first state has only the R action, whereas the last state has only the L action. In any state $s = 1, \ldots, S$, the L action brings the agent to the previous state $s-1$, wheres the R action gets the agent to the next state s with probability p or the same state with probability $1-p$. There is a reward of $+1$ in state S and no reward in any of the other states. The episode ends when the learner reaches state S and we use a discount factor $0 < \gamma \leq 1$.



(i) Compute the value function $V_\gamma(s)$ for the policy that always picks the R action in any state $s = 1, \ldots S$.

(ii) Show that this policy is in fact the optimal policy for this problem.

(iii) What is the optimal average reward $V_\gamma(1)$ for $\gamma = 0$, $\gamma = 1$ and $p = 1$ respectively? Do the results you get make sense?

The definition of the Q-function is $Q(s,a) = r(s) + \gamma \sum_{s'} P(s'|a,s) V_\gamma(s')$, where $P(s'|a,s)$ is the probability of going to state s' when picking action a in state s. Remember that the Q-function Q^* of the optimal policy satisfies the equation $V^*(s) = \max_a Q^*(s,a)$. The idea of Q-learning is to estimate $Q^*(s,a)$ directly from sample transitions (s,a,r,s'). Given an initial estimate $\hat{Q}_{(0)}(s,a)$, we update our estimate according to

$$\hat{Q}_{(i+1)}(s,a) = (1-\alpha)\hat{Q}_{(i)}(s,a) + \alpha\big(r + \gamma \max_{a'} \hat{Q}_{(i)}(s',a')\big) \, . \tag{1}$$

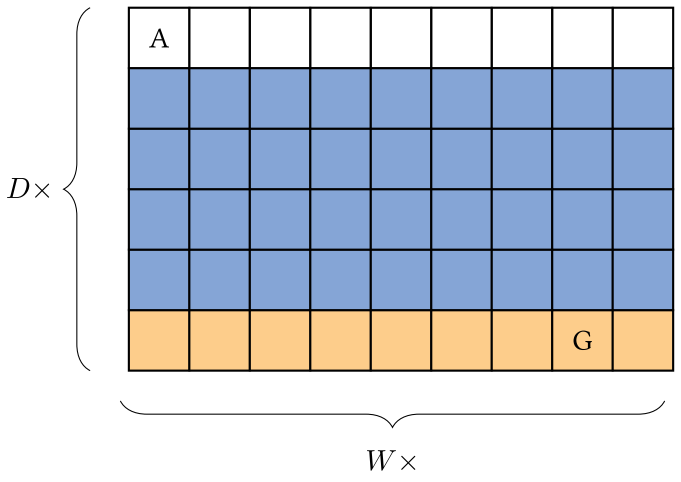
(iv) Suppose you get the following transitions for the MDP above with $S = 3$ and $\gamma = 1$: (1,R,0,2), (2,R,1,3), (1,R,0,1), (1,R,0,2), (2,R,1,3).

Compute the updates for $\hat{Q}_{(i)}$, starting with the initialization $\hat{Q}_{(0)}(s,a) = 0$ for all $s \in \{1,2,3\}$ and $a \in \{\text{L}, \text{R}\}$, and learning rate $0 < \alpha < 1$.

(v) In *online Q-learning*, the idea is to improve the current policy by choosing an action $a_{i+1} = \operatorname{argmax}_a \hat{Q}_{(i)}(s_i, a)$. Assume that we start with the initialization $\hat{Q}_{(0)}(s, \text{L}) = 1$ and $\hat{Q}_{(0)}(s, \text{R}) = 0$ for all states $s = 1, \ldots, S$. What sample trajectory does the policy generate in the MDP above, and does the estimate $\hat{Q}_{(i)}$ converge to the optimal Q-value? Does this agent ever reach state S?

2. Deep Dive

We want to build a learning agent, which dives into the ocean to find a treasures hidden on the seabed. Our problem is modeled the by a grid-world of size $D \times W$ as shown in the figure below. The agent's starting position is denoted by 'A'. At each step, the agent can move to a neighboring cell in any direction using the actions (L), (R), (U), (D) with the obvious limits at the boundary. Transitions are deterministic. The only reward of +1 is given when the agent reaches the treasure 'G' in the bottom row, and its exact position is not known to the agent. We have the additional constraint that our agent can stay below the surface (a non-white cell) for at most T steps, otherwise we receive a reward of -10 and the agent is reset to position 'A'.



(i) The condition that the agent can stay at most T steps below the surface is *non-Markovian*, because it depends on the previous steps taken by the agent. However, by enlarging the state-space we can often make such problems *Markovian*. Find an MDP with $W \times D \times T$ states that models the problem described above, and fully specify its transition model and reward function.

(ii) In order to learn a good policy, the agent needs to gather data by exploration. Perhaps the simplest way of doing exploration is to take an random action in any state. What is the main concern with this exploration strategy in this example, in particular if $T \approx 2D$ and D is very large?

Suppose we run episodes of length $(W + 2T)$. We define the worst-case *sample complexity* of this problem as the maximum expected number of episodes an agent needs to obtain a reward of +1 for the first time in any configuration of the environment. The expectation is over the randomness of the agent.

(iii) Show that the worst-case sample complexity of the random agent is lower bounded by 4^{W+2D-3}. To do so, you need to specify one instance of the environment, such that the expected number of episodes the random agent takes to get +1 reward is 4^{W+2D-3}. For simplicity, assume that $T = 2D - 2$, such that the only way of getting to the treasure is starting the dive directly above it. *Hint*: $\sum_{i=1}^{\infty} i x^{i-1} = \frac{1}{(1-x)^2}$ for $0 < x < 1$.

continues next page...

In the lecture, you have seen the Rmax algorithm, which chooses an *optimistic* initialization of the environment to achieve efficient exploration. Inspired by this idea, we use the following algorithm: First, we initialize an estimate of the reward function $\hat{R}(s) = 1$ for all states s in our MDP. In the ith episode, we then compute the optimal policy in the MDP with the current estimate of reward function $\hat{R}$, and follow this policy for the whole episode. We then use the data from all visited states to update our reward function (the reward is deterministic).

(iv) Show that the worst-case sample complexity of this agent is upper bounded by $W \cdot D \cdot T$. To do so, you need to show that for any instance of the environment, the agent takes at most $W \cdot D \cdot T$ episodes to achieve a reward of +1.

(v) *(Bonus question.)* Assume we want to avoid a negative reward at all cost. Can you find a different initialization, such that our Rmax algorithm still finds the treasure, but never stays below the surface for more than T steps?