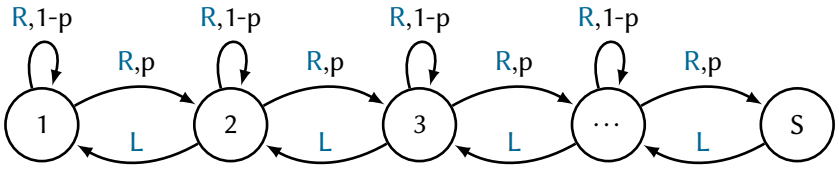
Probabilistic Foundations of Artificial Intelligence

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 $a \in \{\mathrm{L}, \mathrm{R}\}$, 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 $\hat{Q}_{(0)}(s, \mathrm{L}) = 1$ and $\hat{Q}_{(0)}(s, \mathrm{R}) = 0$ 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?

Solutions

(i) In the final state S, we have $V_\gamma(S) = 1$ which follows directly from the definition of the value function. We make use of the Bellman equation for the value function to propagate the value to the previous states. For $S-1$ we get

$$V_\gamma(S-1) = R(S-1) + \gamma\big(pV_\gamma(S) + (1-p)V_\gamma(S-1)\big)\,.$$

Note that $R(S-1) = 0$. Solving for $V_\gamma(S-1)$ yields

$$V_\gamma(S-1) = \frac{\gamma p}{1-\gamma(1-p)} V_\gamma(S)\,.$$

Propagating further yields for any state $s = 1, \dots, S$,

$$V_\gamma(s) = \left(\frac{\gamma p}{1-\gamma(1-p)}\right)^{S-s} V_\gamma(S)\,.$$

Remark: In general, we can solve the linear system given by the Bellman equation, ie $V = R + \gamma PV$, where $V \in \mathbb{R}^S$ is the value function and $R \in \mathbb{R}^d$ is the reward vector.

(ii) A policy is optimal if and only if it is greedy w.r.t. its induced value function. We check if the policy chooses $a \in \operatorname{argmax}_{a'} Q(s,a')$ for $Q(s,a') = R(s) + \gamma \sum_{s'} p(s'|a',s)V_\gamma(s')$ in any state $s < S$. We get

$$\begin{aligned} Q(s,\mathrm{L}) &= 0 + \gamma V_\gamma(s-1)\,,\\ Q(s,\mathrm{R}) &= 0 + \gamma(pV_\gamma(s+1) + (1-p)V_\gamma(s))\,. \end{aligned}$$

With the previously computed values for V_γ, it is clear, that R is the greedy action. Therefore the policy is optimal.

(iii) For $\gamma = 0$ we get $V_0(1) = 0$. $\gamma = 0$ essentially makes the agent not care about future rewards. For $\gamma = 1$, we get $V_\gamma(0) = 1$. Since there is no discounting, it doesn't matter how long the agent takes to reach state S. Finally, if $p = 1$, the R action becomes deterministic. We have $V_\gamma(1) = \gamma^{S-1}$. This we could have gotten directly from the definition of $V_\gamma(1) = \sum_{i=0}^{S} \gamma^i R(s_i)$.

(iv) We initialize for all $s \in \{1,2,3\}$, $a \in \{\mathrm{L},\mathrm{R}\}$,

$$\hat{Q}_{(0)}(s,a) = 0$$

Update on (1,R,0,2):

$$\begin{aligned} \hat{Q}_{(1)}(1,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(0)}(1,\mathrm{R}) + \alpha(0 + \max_{a'} \hat{Q}_{(0)}(2,a'))\\ &= (1-\alpha)0 + \alpha 0 = 0 \end{aligned}$$

Update on (2,R,1,3):

$$\begin{aligned} \hat{Q}_{(2)}(2,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(1)}(2,\mathrm{R}) + \alpha(1 + \max_{a'} \hat{Q}_{(1)}(3,a'))\\ &= (1-\alpha)0 + \alpha 1 = \alpha \end{aligned}$$

Update on (1,R,0,1):

$$\begin{aligned}\hat{Q}_{(3)}(1,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(2)}(1,\mathrm{R}) + \alpha(0 + \max_{a'} \hat{Q}_{(2)}(1,a')) \\ &= (1-\alpha)0 + \alpha 0 = 0\end{aligned}$$

Update on (1,R,0,2):

$$\begin{aligned}\hat{Q}_{(4)}(1,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(3)}(1,\mathrm{R}) + \alpha(0 + \max_{a'} \hat{Q}_{(3)}(2,a')) \\ &= (1-\alpha)0 + \alpha(0+\alpha) = \alpha^2\end{aligned}$$

Update on (2,R,1,3):

$$\begin{aligned}\hat{Q}_{(5)}(2,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(4)}(2,\mathrm{R}) + \alpha(1 + \max_{a'} \hat{Q}_{(4)}(3,a')) \\ &= (1-\alpha)\alpha + \alpha(1+0) = \alpha(2-\alpha)\end{aligned}$$

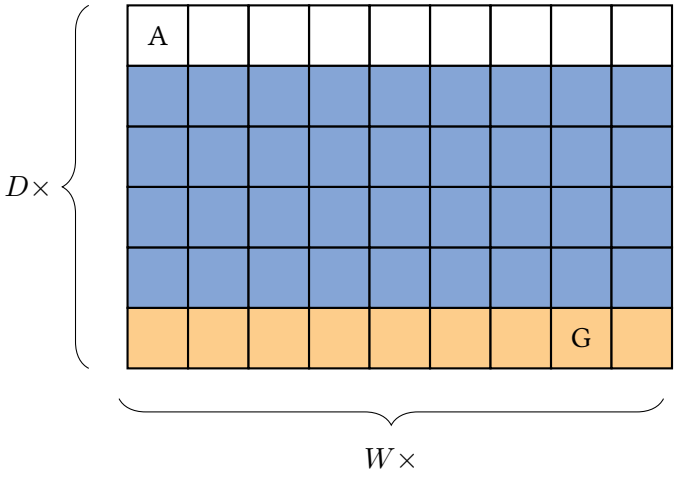
(v) We have initial values $\hat{Q}_{(0)}(s,\mathrm{L}) = 1$ and $\hat{Q}_{(0)}(s,\mathrm{R}) = 0$. Before we do the calculations, note that if we act greedily w.r.t. this Q-function, we always prefer the L action over R. This is problematic, because using this policy, we will never reach state S, where we get a reward. Formally, starting from $s = 1$, there is only the R action. Therefore we can get the transition $(1,\mathrm{R},0,1)$ or $(1,\mathrm{R},0,2)$. The former does not affect our estimate $\hat{Q}_{(i)}$ because $\hat{Q}_{(0)}(s,\mathrm{R}) = 0$ and there is on Laction. For the case where we transit to state $s = 2$, we update

$$\begin{aligned}\hat{Q}_{(i+1)}(1,\mathrm{R}) &= (1-\alpha)\hat{Q}_{(i)}(1,\mathrm{R}) + \alpha(0 + \max_{a'} \hat{Q}_{(3)}(2,a')) \\ &= (1-\alpha)\hat{Q}_{(i)}(1,\mathrm{R}) + \alpha(0+1) = (1-\alpha)\hat{Q}_{(i)}(1,\mathrm{R}) + \alpha < 1\end{aligned}$$

the last inequality follows inductively from the the initialization $\hat{Q}_{(i)}(1,\mathrm{R}) = 0$. After this update, the Laction is still better, therefore the agent goes back to state 0. From this we see that initialization can matter a lot for exploration. Often, an optimistic initialization, e.g. $\hat{Q}_{(0)}(s,\mathrm{L}) = 1$ and $\hat{Q}_{(0)}(s,\mathrm{R}) = 1$, avoids problems like this.

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+2T-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?

Solutions

(i) We define an extended state space by $S = \{(x, y, t) : x = 1, \dots, W; y = 1, \dots, D; t = 0, \dots T\}$. The additional dimension allows to keep track of how long the agent stays below the surface. The starting state is $(1, 1, 0)$ We define the transitions as follows.

In any state $(x, 1, 0)$, for $x = 1, \dots, W$, we have

$$\begin{aligned} L &\rightsquigarrow (x-1, 1, 0) && \text{(moving above the surface)} \\ R &\rightsquigarrow (x+1, 1, 0) && \text{(moving above the surface)} \\ D &\rightsquigarrow (x, 2, 1) && \text{(going below the surface)} \\ U &\rightsquigarrow (x, 1, 0) && \text{(boundary)} \end{aligned}$$

In any state $(x, 2, t)$, for $x = 1, \dots, W$, and $t = 1, \dots T$, we have

$$\begin{aligned} L &\rightsquigarrow (x-1, 2, t+1) \\ R &\rightsquigarrow (x+1, 2, t+1) \\ D &\rightsquigarrow (x, 3, t+1) \\ U &\rightsquigarrow (x, 1, 0) && \text{(reaching the surface)} \end{aligned}$$

In any state (x, y, t), for $x = 1, \dots, W$, $y = 3, \dots, D$ and $t = 1, \dots T$, we have

$$\begin{aligned} L &\rightsquigarrow (x-1, y, t+1) \\ R &\rightsquigarrow (x+1, y, t+1) \\ D &\rightsquigarrow (x, y+1, t+1) \\ U &\rightsquigarrow (x, y-1, t+1) \end{aligned}$$

And finally, for any (x, y, T) any action leads to $(1, 1, 0)$ (reset).

The reward function is $R(x, y, T) = -10$ for all $x = 1, \dots W$ and $y = 1, \dots, D$. If the goal is at (x^*, y^*) in the 2d grid world, we set the reward of $R(x^*, y^*, t) = +1$ for any $t < T$. All other states have zero reward.

Remark 1: By our transition model, not every state in S is reachable, e.g. the state $(1, D, 1)$ for $D > 2$ cannot be reached.

Remark 2: There is a catch the way the reward is generated in our extended state-space: If an agent picks up the treasure at (x^*, y^*, t) it can get an additional reward by going to a neighbouring cell and then back to the reward cell at $(x^*, y^*, t + 2)$ to get another reward of +1. To avoid this, one would have to introduce a further flag to keep track of whether the goal has been reached or not. This can be done by using an extended state space (x, y, t, f) where f indicates if the agent was at the position (x^*, y^*).

Remark 3: By using the whole history as state, it is possible to convert very general reinforcement settings to an MDP. However this comes at the cost of a state-space which is exponentially larger than the original state space.

(ii) The random exploration strategy might take a very long time until it discovers the position of the treasure. In addition it does not actively avoid the cost of staying below the surface for too long - something which is very likely to happen when doing random exploration.

(iii) We place the treasure in the bottom-right corner of the grid world. For $T = 2D - 2$, the only path that leads to the treasure and avoids the -10 reward, is

$$\underbrace{\text{R}, \ldots, \text{R}}_{W-1\times}, \underbrace{\text{D}, \ldots, \text{D}}_{\text{D-1}\times}, \underbrace{\text{U}, \ldots, \text{U}}_{\text{D-1}\times} .$$

In any episode, the probability that the agent takes this sequence of actions is $p = \left(\frac{1}{4}\right)^{W+2D-3}$. The expected number of episodes needed is therefore

$$\mathbb{E}[\#\text{episodes}] = \sum_{i=1}^{\infty} pi(1-p)^{i-1} = \frac{p}{(1-(1-p))^2} = \frac{1}{p} = 4^{W+2D-3} .$$

(iv) Each time the optimistic policy visits a state (x, y, t) with no reward, it updates the reward estimate $\hat{R}(x, y, t) = 0$. Consequently in the next episodes, the updated policy (which is optimal for the estimated reward $\hat{R}$) needs to visit at least one new state (since for unvisited states, the reward estimate is still +1). Because there are only that many states, after $W \cdot D \cdot T$ episodes, the Rmax algorithm has found the treasure.

(v) If we set $\hat{R}(x, y, T) = -10$ from the beginning, the Rmax algorithm avoids staying below the surface for too long, since any policy which stays above the surface has reward of 0. If we additionally set all other states $\hat{R}(x, y, t) = +1$, we still get the same exploration incentive as in the previous part.