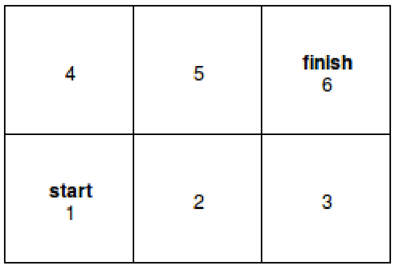
Answers Mid-Course Test Reinforcement Learning

Artificial Intelligence Techniques (IN4010)

December 21^{st}, 2016

Assume we are an agent in a 3x2 gridworld, as shown in the below figure. We start at the bottom left node (1) and finish in the top right node (6). When node 6 is reached, we receive a reward of +10 and return to the start for a new episode. On all other actions that not lead to state 6, the reward is -1.



In each state we have four possible actions: up, down, left and right. For each action we move deterministically in the specific direction on the grid. Assume that we cannot take actions that bring us outside the grid.

The current estimates of $Q(s, a)$ are given in the below table:

Q(1,up)=4			Q(1,right)=3
Q(2,up)=6		Q(2,left)=3	Q(2,right)=8
Q(3,up)=9		Q(3,left)=7	
	Q(4,down)=2		Q(4,right)=5
	Q(5,down)=6	Q(5,left)=5	Q(5,right)=8

Question a (1p) Since we have full environmental knowledge, we can apply Bellman's equation to further update the Q estimates (i.e. dynamic programming). We take a **greedy** policy and $\gamma = 0.9$. For convenience, the Q-specification of Bellman's equation is given (where s' and a' denote the next state and action, respectively):

$$Q(s, a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \sum_{a'} \pi(s', a') Q(s', a')] \qquad (1)$$

Perform a single update of Q(3,left).

Answer: Q(3,left) = 1 [-1 + 0.9 * (1 * 8)] = 6.2*

Note that because the environment is deterministic, we only need to sum over one s' (which has $P_{3,2}^{left} = 1$), and because the policy is greedy, we only sum over one next action a' (which has probability $\pi(2, right) = 1$.

Question b (1p) We now no longer assume a model of the environment. The above table was rather created through temporal difference learning, where we sample through the state-space. Why is it not smart to take the greedy policy now (from the start)? What should be balanced here?

Answer:

- *(0.5p) In the sampling-based/learning setting, the greedy policy may lead to suboptimal results, since we do not ensure exploration. Maybe there are parts of the state-space with even better rewards, but we just have not visited them yet.*
- *(0.5p) What we should balance is exploration versus exploitation. The greedy policy only considers exploitation.*

Question c (1p) Why were we using Q-values? What is the advantage of learning state-action values (Q) compared to state values (V)? *(Hint: consider action selection)*

Answer: The advantage is in the action selection. With Q-values, we can directly see the value of each available action, and use these in e.g. ϵ-greedy or softmax action selection. On the contrary, when we only have V-estimates, we need to 1) do extra calculations, and 2) have a transition model ($P_{ss'}^a$), so at each decision moment we can calculate the $Q(s, a) = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')]$ for all available actions a. Especially when controlling a real-time system, your decisions have to be fast.

Question d (1p) We decide to switch to **softmax** exploration:

$$\pi(s, a) = \frac{e^{Q(s,a)}}{\sum_b e^{Q_(s,b)}} \tag{2}$$

We are currently in node 2. Give the probability that we will move right on the next step (you can write the equation with correct numeric values, but you may skip calculating the resulting decimal number).

Answer: $\pi(2, right) = \frac{e^8}{e^8+e^6+e^3}$

Dont forget to sum over all available actions in the denominator, including the one you are calculating the probability for. I see some of you omitting the e^8 *term in the denominator in this example. However, then the probability distribution over all actions will not sum up to 1.*

Question e (1p) We will continue updating the Q-table with a SARSA (state-action-reward-state-action) algorithm. Starting from node 2, we have sampled the following trajectory: 2 - up - 5 - right - 6, after which the trial ended. Update two $Q(s, a)$ entries by filling in the form below (take $\alpha = 0.2, \gamma = 0.8$) . The SARSA equation is provided:

$$Q(s, a) = Q(s, a) + \alpha[R^a_{ss'} + \gamma Q(s', a') - Q(s, a)] \quad (3)$$

Answer:

s	*a*	*r*	*s'*	*a'*
2	*up*	*-1*	*5*	*right*

*Q(1, up) = 6 + 0.2 * [-1 + 0.8 * 8 - 6] = 6 - 0.12 = 5.88*

s	*a*	*r*	*s'*	*a'*
5	*right*	*10*	*6*	*- (episode terminated)*

*Q(5, right) = 8 + 0.2 * [10 + 0.8 * 0 - 8] = 8 + 0.4 = 8.4*

Note that the (action)-value of a terminal state is 0. This is important in the second calculation.