Prof. Dr. techn. Wolfgang Nejdl
Franziska Schoger

November 29th 2023

Advanced Topics in AI
Exercise 5

Question 1: Unknown Games

Imagine an unknown game. Suppose a game agent chooses actions according to some policy π and generates a sequence of actions and rewards in the unknown game. *Unless specified otherwise, assume a discount factor* $\gamma = 0.5$ *and a learning rate* $\alpha = 0.5$

a. The game is split in three episodes:

t	s_t	a_t	s_{t+1}	r_t
Episode 1				
0	A	Down	B	2
1	B	Down	B	-4
2	B	Down	C	0
3	C	Exit	x	10
Episode 2				
0	A	Down	A	-1
1	A	Down	B	2
2	B	Down	C	0
3	C	Exit	x	10
Episode 3				
0	A	Down	B	2
1	B	Down	B	-4
2	B	Down	C	0
3	C	Exit	x	10

i. In model-based reinforcement learning, we first estimate the transition function $T(s, a, s')$ and the reward function $R(s, a, s')$. Fill in the following estimates of T and R, estimated from the experience above. Write "n/a" if not applicable or undefined.

$\hat{T}(A, Up, A) =$ $\hat{T}(A, Down, B) =$ $\hat{T}(B, Down, C) =$ $\hat{T}(C, Exit, x) =$

$\hat{R}(A, Up, A) =$ $\hat{R}(A, Down, B) =$ $\hat{R}(B, Down, C) =$ $\hat{R}(C, Exit, x) =$

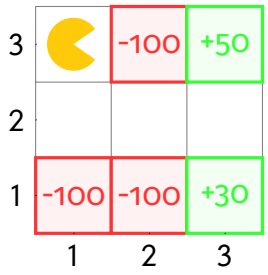
ii. In model-free reinforcement learning, we don't need the transition and reward function. Assume the experience given in (a). Using direct evaluation, calculate the following values:

$V(\text{A}) =$ $V(\text{B}) =$ $V(\text{C}) =$

b. If we repeatedly feed this experience sequence through our Q-learning algorithm, what values will it converge to? Assume the learning rate α_t is properly chosen so that convergence is guaranteed.
   - the values $\hat{V}^*$ (the optimal values corresponding the estimates of T and R)
   - the optimal values V^*
   - neither $\hat{V}^*$ nor V^*
   - not enough information to determine

Question 2: Q-Learning in Gridworld

Consider the grid-world given below and an agent who is trying to learn the optimal policy. Rewards are only awarded for taking the Exit action from one of the shaded states. Taking this action moves the agent to the Done state and the MDP terminates. Assume $\gamma = 1$ and $\alpha = 0.5$ for all calculations.



The agent starts from the top left corner and you are given the following episodes from runs of the agent through this grid-world. Each line in an Episode is a tuple containing (s, a, s_0, r).

Episode 1	Episode 2	Episode 3	Episode 4	Episode 5
(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0	(1,3), S, (1,2), 0
(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0	(1,2), E, (2,2), 0
(2,2), E, (3,2), 0	(2,2), S, (2,1), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0	(2,2), E, (3,2), 0
(3,2), N, (3,3), 0	(2,1), Exit, D, -100	(3,2), S, (3,1), 0	(3,2), N, (3,3), 0	(3,2), S, (3,1), 0
(3,3), Exit, D, +50		(3,1), Exit, D, +30	(3,3), Exit, D, +50	(3,1), Exit, D, +30

Q-learning is an online algorithm to learn optimal Q-values in an MDP with unknown rewards and transition function. The update equation is:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

where γ is the discount factor, α is the learning rate and the sequence of observations are $(\ldots, s_t, a_t, s_{t+1}, r_t, \ldots)$. All Q-values are initialized as 0.

a. Given the episodes above, fill in the time at which the following Q values first become non-zero. Your answer should be of the form (episode#,iter#) where iter# is the Q-learning update iteration in that episode. If the specified Q value never becomes non-zero, write never.

   $Q((1,2), \text{E})$: $Q((2,2), \text{E})$: $Q((3,2), \text{S}))$:

b. What are the learned Q-values from Q-learning after all five episodes?

Question 3: Pacman with Feature-Based Q-Learning

We would like to use a Q-learning agent for Pacman, but the state size for a large grid is too massive to hold in memory. To solve this, we will switch to feature-based representation of Pacman's state.

a. Say our two minimal features are the number of ghosts within 1 step of Pacman (F_g) and the number of food pellets within 1 step of Pacman (F_p). You'll notice that these features depend only on the state, not the actions you take. Keep that in mind as you answer the next couple of questions. For this pacman board:

   Extract the two features (calculate their values).

b. With Q Learning, we train off of a few episodes, so our weights begin to take on values. Right now $w_g = 100$ and $w_p = -10$. Calculate the Q value for the state above.

c. We receive an episode, so now we need to update our values. An episode consists of a start state s, an action a, an end state s' , and a reward r. The start state of the episode is the state above (where you already calculated the feature values and the expected Q value). The next state has feature values $F_g = 0$ and $F_p = 2$ and the reward is 50. Assuming a discount of $\gamma = 0.5$, calculate the new estimate of the Q value for s based on this episode.

d. With this new estimate and a learning rate (α) of 0.5, update the weights for each feature.

Question 4: Policy Evaluation

In this question, you will be working in an MDP with states S, actions A, discount factor γ, transition function T, and reward function R.

We have some fixed policy $\pi : S \rightarrow A$, which returns an action $a = \pi(s)$ for each state $s \in S$. We want to learn the Q function $Q^{\pi}(s, a)$ for this policy: the expected discounted reward from taking action a in state s and then continuing to act according to $\pi : Q^{\pi}(s, a) = \Sigma_{s'} T(s, a, s')[R(s, a, s') + \gamma Q^{\pi}(s', \pi(s'))]$. The policy π will not change while running any of the algorithms below.
Suppose T and R *are unknown*. You will develop sample-based methods to estimate Q^{π}. You obtain a series of samples $(s_1, a_1, r_1), (s_2, a_2, r_2)...(s_T, a_T, r_T)$ from acting according to this policy (where $a_t = \pi(s_t)$, for all t).
Recall the update equation for the Temporal Difference algorithm, performed on each sample in sequence:

$$V(s_t) \leftarrow (1 - \alpha)V(s_t) + \alpha(r_t + \gamma V(s_{t+1}))$$

which approximates the expected discounted reward $V^{\pi}(s)$ for following policy π from each state s, for a learning rate α.

a. Fill in the blank below to create a similar update equation which will approximate Q^{π} using the samples. You can use any of the terms $Q, s_t, s_{t+1}, a_t, a_{t+1}, r_t, r_{t+1}, \gamma, \alpha, \pi$ in your equation, as well as $\sum$ or max but no other terms.

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(\qquad\qquad)$$

b. Now, we will approximate Q^{π} using a linear function: $Q(s, a) = \Sigma_{i=1}^{d} w_i f_i(s, a)$ for weights $w_1, ..., w_d$ and feature functions $f_1(s, a), ..., f_d(s, a)$.
To decouple this part from the previous part, use Q_{samp} for the value in the blank in part (i) (i.e. $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha Q_{samp}$)
Which of the following is the correct sample-based update for each w_i ?

- $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]$
- $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]$
- $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]f_i(s_t, a_t)$
- $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]f_i(s_t, a_t)$
- $w_i \leftarrow w_i + \alpha[Q(s_t, a_t) - Q_{samp}]w_i$
- $w_i \leftarrow w_i - \alpha[Q(s_t, a_t) - Q_{samp}]w_i$

c. The algorithms in the previous parts are:

- model-based
- model-free