NAME:_____

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA
Dept. Teoria del Senyal i Comunicacions

230817-ARAP
9-11-2020
Duration: 1h 45'

Exercise 1 (Multi-Armed Bandits)

Consider a k-armed bandit problem with k=4 actions, denoted as 1, 2, 3, 4. We apply to this problem an ε-greedy action selection and an estimation of the action values $Q(a)$ based on average of the rewards with initial random values $Q_1(1)$=0.1, $Q_1(2)$=-0.1, $Q_1(3)$=0.2, $Q_1(4)$=0.01. Suppose the initial sequence of actions A and rewards R is:

A_1=1, R_1=-1, A_2=2, R_2=-1, A_3=2, R_3=2, A_4=2, R_4=-2, A_5=3, R_5=0, A_6=4, R_6=1

In some of these time steps the ε case may have occurred, causing an action to be selected at random. Identify these steps and support your answer.

We just need to evaluate the update of the value function over time for each action using for example the recursive expression:

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{t_a}\left(R^{(a)} - Q_t(a)\right)$$

Using the episode provided above, we obtain the following set of values:

	t=1		t=2		t=3		t=4		t=5		t=6	
	R	Q	R	Q	R	Q	R	Q	R	Q	R	Q
Q(1)	-1	1		1		1		1		1		1
Q(2)		-0.1	-1	-1	2	-2.5	-2	-2.65		-0.33		-0.33
Q(3)		0.2		0.2		0.2		0.2	0	0		0
Q(4)		-0.01		0.01		0.01		0.01		0.01	1	1

From the observation of actions selected, the greedy strategy is only adopted in time instants 3, 4, 5 and 6.

NAME:_____

Exercise 2 (Temporal difference learning)

We are asked to solve a reinforcement learning task using time difference methods. In order to have more exploratory behavior we propose to use Q-learning as an off-policy strategy (evaluate one policy while following another), even though the convergence can be slower.

a) Complete the missing terms in the action-value update, both for SARSA and Q-learning:

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state,\cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \quad - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

For SARSA: $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$

For Q-learning: $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

b) If adopting SARSA with a greedy selection of the action ($\varepsilon = 0$), would it take the same actions and compute the same Q(S,A) updates as the Q-learning method with $\varepsilon \neq 0$? Why?
The selection of actions A' may be different: SARSA with $\varepsilon = 0$ takes greedy actions, while Q-learning with $\varepsilon \neq 0$ explores. Therefore, S' may also be different. Regarding the update of Q(S,A), both would use the maximum of Q(S',A'), but neither the value of R or the maximum value of Q(S',A') would be the same because of the different selection of actions A'.

c) In order to improve the performance of Q-learning we decide to switch to expected-SARSA. What is the update equation for action-value function? What are the benefits?

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \sum_{a \in \mathcal{A}} \pi(a|S')Q(S',a) - Q(S,A)\right]$$

The benefit comes from a reduction in the variance in the estimation of Q(S,A), associated to the selection of A'. The expected-SARSA algorithm updates Q(S,A) using an average of Q(S',A) values obtained from SARSA decisions in state S'. Therefore expected-SARSA moves in the direction of SARSA moves in expectation. In all cases, the policy followed by the agent π can be an ε-greedy policy.

d) Can it be considered an off-policy (and hence more exploratory) method? Why?
Actions are taken using a certain policy (e.g. ε-greedy) from the updated values of Q(S,A), but the update is computed using an average of values, not only focusing on the adopted action. Then it is off-policy.

NAME:_____

Exercise 3 (MDP and Bellman)

The state value function $v(s)$ of a Markov Reward Process (MRP) is the expected return starting from state s: $v(s) = R_s + \gamma \sum_{s' \in S} p(s'|s)v(s')$.

In the store of a pharmacy there is capacity to keep a maximum of 50 units of a very demanded drug. Every day the number of sold units (n) follows a Poisson distribution of mean equal to 5, i.e, $\Pr\{n\} = \frac{\lambda^n e^{-\lambda}}{n!}$; $\lambda = 5$. The profit for each dispatched unit is 2€. At night, the pharmacist can order a batch of 10 units for a total cost of 15€, order a batch of 30 units for a total cost of 30€ or keep the current stock. If he orders more units than he can retain that day, he can only keep the maximum number without exceeding 50, but he still has to pay the total amount ordered.

Answer next questions:

a) Taking a day as the time step, define a Markov Decision Process (MDP) such that the state is defined by the stock of drug units at the closing time. Identify the space of actions that the pharmacist can do once the pharmacy is closed. Write the Bellman equation system in matrix form for the MRP obtained when you follow a policy π, give the size of all vectors and matrices of the system.

Space of Actions: $\mathcal{A} = \{0,10,30\}$ where the action is the number of ordered units.

Space of states: $\mathcal{S} = \{0,1,2, \dots ,50\}$ where the state is the stock of drug units at the closing time and $|\mathcal{S}|$=51

$(P - \gamma I)r^{\pi} = v_{\pi}$ where P is the $|\mathcal{S}|x|\mathcal{S}|$ transition probability matrix, γ is the discount factor, r^{π} is the $|\mathcal{S}|$x1 averaged reward vector and v_{π} is the state-value function $|\mathcal{S}|$x1 vector.

b) If the pharmacist follows a uniform policy to choose the action, compute the transition probability $\Pr\{s_{t+1} = 25|s_t = 20\}$.

$\Pr\{s_{t+1} = 25|s_t = 20\}$=$\pi(a_t = 0|s_t = 20)$ $\Pr\{s_{t+1} = 25|s_t = 20, a_t = 0\}$+ $\pi(a_t = 10|s_t = 20)$ $\Pr\{s_{t+1} = 25|s_t = 20, a_t = 10\}$+ $\pi(a_t = 30|s_t = 20)$ $\Pr\{s_{t+1} = 25|s_t = 20, a_t = 30\}$

Following a uniform policy: $\pi(a_t|s_t) = \frac{1}{3} \;\; \forall s_t \in \mathcal{S}; \forall a_t \in \mathcal{A}$, so:

$\Pr\{s_{t+1} = 25|s_t = 20\}$=$\frac{1}{3}$($0 + \frac{5^5 e^{-5}}{5!} + \frac{5^{25} e^{-5}}{25!}$)

c) If the pharmacist follows a uniform policy to choose the action, compute the averaged reward $r(s = 5)$.

$$r^{\pi}(s) = \sum_{a_t \in \mathcal{A}} \pi(a_t|s_t) \sum_{s_{t+1} \in \mathcal{S}} \Pr\{s_{t+1}|s_t, a_t\} r(s_t, a_t, s_{t+1}) \Rightarrow r^{Uniform}(5) =$$

$$\frac{1}{3}\left(\sum_{n=0}^{5} \frac{5^n e^{-5}}{n!} 2n + \sum_{n=6}^{\infty} \frac{5^n e^{-5}}{n!} 10\right) + \frac{1}{3}\left(-15 + \sum_{n=0}^{15} \frac{5^n e^{-5}}{n!} 2n + 30 \sum_{n=16}^{\infty} \frac{5^n e^{-5}}{n!}\right)$$

$$+\frac{1}{3}\left(-30 + \sum_{n=0}^{35} \frac{5^n e^{-5}}{n!} 2n + 70 \sum_{n=36}^{\infty} \frac{5^n e^{-5}}{n!}\right)$$

d) How many deterministic policies there exist for this MDP? $|\mathcal{A}|^{|\mathcal{S}|} = 3^{51}$

NAME:_____

Exercise 4 (Dynamic Programming)

In Control Dynamic Programming the non-linear Bellman optimality equation is solved by iterating between the improvement of the policy ($\pi(s|a)$) and the evaluation of the value function associated to the policy ($v_\pi(s)$). This is the principle of the two methods described in class: Policy Iteration (PI) and Value Iteration (VI).

a) Explain if PI and VI are model-based or model-free algorithms. Model based, Transition probabilities are known.

b) Both of them (PI and VI) follow a general policy iteration (GPI) methodology. Which is the main difference between them? In PI policy is evaluated until convergence and then it is improved while in VI policy is evaluated only with single iteration and then it is improved.

c) In the following you can observe an algorithm code: Which of the two methods (PI or VI) is implemented by this code? PI

d) Write near the main lines of the code the formulas or instructions they implement.

```
while ac_dif > 0 and iter < it_max + 1:
    ac_dif = 0
    actions = [] # Store actions taken in each state in one run.
    policy_evaluation(grid)given π, vᵖ(s) is iteratively computed
    for s in range(1, grid.N_STATES + 1):
        action_taken = grid.POLICY[s-1].argmax()
        action_values = np.zeros(grid.N_ACTIONS)
       for a in range(grid.N_ACTIONS):
            next_state, reward = grid.next_position(s, a)
            action_values[a] += reward + grid.GAMMA*grid.V_FUNCTION[next_state-1]
        best_action = action_values.argmax()π(s) ← argmax_a Σ_{s',r} Pr(s',r|s,a) [r + γV(s')]
                                                                    = argmax_a(Q(s,a))
        if action_taken != best_action:
            ac_dif += 1 count the number of states which changes best action
        grid.POLICY[s-1] = np.zeros(grid.N_ACTIONS)
        grid.POLICY[s-1][best_action] = 1 π(a|s)
        actions.append(np.argmax(grid.POLICY[s-1]))
      actions_list.append(np.transpose(np.array(actions).reshape(grid.N_ROWS,grid.N_COLUMNS)))
    a_inc.append(ac_dif)
    iter += 1
```