

ROB311 Quiz 3

Hanhee Lee

March 20, 2025

Contents

1	Reinforcement Learning	2
1.1	Estimating Q-Star Empirically	2
1.1.1	Running Average Update Rule	2
1.2	Q-Learning Algorithm	3
1.3	Modified Q-Learning Algorithm	3
1.4	Training vs. Testing	4
1.4.1	K Sims, 1 Test	4
1.4.2	K Tests	4
2	Partially Observable MDPs (POMDPs)	5
2.1	Bayesian Network	5
3	Multi-Agent Problems	8
3.1	Action Equilibria	8
3.1.1	Finding Action Equilibria	8
3.2	Strategy Equilibria	8
3.2.1	Finding Strategy Equilibria	8
3.2.2	Existence of Strategy Equilibria	8
3.2.3	Convergence of Strategy Equilibria	8
3.3	Examples	8
3.3.1	Optimal Action Profiles	8

Partially Observable Probabilistic Decision Problems

1 Reinforcement Learning

Summary: In a RL problem, $p(\cdot | \cdot, \cdot)$ and/or $r(\cdot, \cdot)$ unknown.

1.1 Estimating Q-Star Empirically

Summary:

γ	Equation
0	$q^*(s, a) = \lim_{K \rightarrow \infty} \bar{R}_K$ <ul style="list-style-type: none"> $\bar{R}_K = \frac{1}{K} \sum_{k=1}^K r_k$: empirical average reward. r_k: reward obtained in the k^{th} simulation. K: # of times action a taken in state s (# of simulations)
0	$q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} (r(s, a, s') - q^*(s, a))$ <ul style="list-style-type: none"> $N(s, a)$: # of times action a taken in state s.
$\neq 0$	$q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left(\left[r(s, a, s') + \gamma \max_{a'} q^*(s', a') \right] - q^*(s, a) \right)$ <ul style="list-style-type: none"> Using old q^* values to estimate.

1.1.1 Running Average Update Rule

Definition:

$$\bar{x} \leftarrow \bar{x} + \alpha(x_{\text{new}} - \bar{x}).$$

- α : learning rate

1.2 Q-Learning Algorithm

Algorithm:

```

1 procedure Q_LEARNING():
2   for each episode do
3     set initial state  $s \leftarrow s_0$ 
4     while  $s \notin \mathcal{T}$  do #  $\mathcal{T}$ : terminal states
5       randomly choose an action in  $\mathcal{A}(s)$ 
6       get next state,  $s'$ , and reward  $r$ 
7       update  $N(s, a)$  and  $q^*(s, a)$  as follows:
8
9        $q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left( r(s, a, s') + \gamma \max_{a'} q^*(s', a') - q^*(s, a) \right)$ 
10
11        $N(s, a) \leftarrow N(s, a) + 1$ 
12
13        $s \leftarrow s'$ 
14     end while
15   end for

```

- **Note:** Possible infinite while loop if \mathcal{T} is not reached.

1.3 Modified Q-Learning Algorithm

Algorithm:

```

1 procedure Q_LEARNING():
2   for each episode do
3      $l \leftarrow 0$ 
4     set initial state  $s \leftarrow s_0$ 
5     while  $s \notin \mathcal{T}$  and  $l < l_{\max}$  do
6       randomly choose an action in  $\mathcal{A}(s)$ 
7       get next state,  $s'$ , and reward  $r$ 
8       update  $N(s, a)$  and  $q^*(s, a)$  as follows:
9
10       $q^*(s, a) \leftarrow q^*(s, a) + \frac{1}{N(s, a)} \left( r(s, a, s') + \gamma \max_{a'} q^*(s', a') - q^*(s, a) \right)$ 
11
12       $N(s, a) \leftarrow N(s, a) + 1$ 
13
14       $l \leftarrow l + 1$ 
15       $s \leftarrow s'$ 
16    end while
17  end for

```

Notes: Choice of γ and l_{\max} are coupled:

- $\gamma \approx 1$ requires large l_{\max}
- $\gamma \approx 0$ requires small l_{\max}

1.4 Training vs. Testing

Notes: Episodes are classified as either:

- training (sim): reward accumulated during episode does not count
- testing (test): reward accumulated during episode counts

1.4.1 K Sims, 1 Test

Notes:

1. select actions randomly during K simulations
2. extract optimal policy, π^*
3. use π^* during test

1.4.2 K Tests

Notes:

- maximize average reward over K tests
- must balance between exploration and exploitation
- Common ways to balance exploration and exploitation: ε -greedy strategy, UCB algorithm

Strategy	Description
ε -greedy	<p>choose optimal action with probability $\varepsilon(k)$</p> <ul style="list-style-type: none"> • In episode k, choose the optimal action with probability $\varepsilon(k)$, where: <ul style="list-style-type: none"> – $\varepsilon(0) \approx 0$ – $\varepsilon(k)$ is increasing – $\varepsilon(k) \rightarrow 1$ as $k \rightarrow \infty$ • Common choice for $\varepsilon(k)$ is $1 - \frac{1}{k}$.
UCB algorithm	<p>choose action that maximizes $\text{UCB}(\cdot)$</p> $\text{UCB}(s, a) = \begin{cases} q^*(s, a) + C \sqrt{\frac{\log k}{N(s, a)}}, & \text{if } N(s, a) > 0 \\ \infty, & \text{otherwise} \end{cases}$ <ul style="list-style-type: none"> • In episode k, choose the action that maximizes $\text{UCB}(\cdot)$. • C: exploration parameter • $N(s, a)$: # of times a taken from s.

2 Partially Observable MDPs (POMDPs)

Summary: In a POMDPs, we assume that:

- environment modelled using state space, \mathcal{S}
- single agent
- S_t = state after transition t
- A_t = action inducing transition t
- stochastic state transitions with memoryless property:

$$S_T \perp S_0, A_1, \dots, A_{T-1}, S_{T-2} \mid S_{T-1}, A_T$$

- R_t = reward for transition t , i.e., (S_{T-1}, A_T, S_T)
- O_t = observation of S_t

Name	Function:
Initial state distribution	$p_0(s) := \mathbb{P}[S_0 = s]$
Transition distribution	$p(s' s, a) := \mathbb{P}[S_t = s' A_t = a, S_{t-1} = s]$
Reward function	$r(s, a, s') := \text{reward for transition } (s, a, s')$
<ul style="list-style-type: none"> • Since actual state is unknown, so are legal actions. • Can fix by assuming $\mathcal{A}(s) = \mathcal{A}(s') := \mathcal{A} \forall s, s'$: <ul style="list-style-type: none"> – if $a \notin \mathcal{A}(s)$, then $p(s' s, a) = 0$ for all $s' \neq s$ – if $a \notin \mathcal{A}(s)$, then $r(s, a, s') = 0$ for all s' 	
Policy for choosing actions	$\pi_t(a o_0, \dots, o_t) := \mathbb{P}[A_t = a O_0 = o_0, \dots, O_t = o_t]$
Measurement model	$m(o s) := \mathbb{P}[O_t = o S_t = s]$
<ul style="list-style-type: none"> • Observe that policy is now time-dependent. • Special Case: If we assume the agent cannot use past observations, $A_t \perp O_0, \dots, O_{t-1} \mid O_t$, policy becomes time-independent, $\pi_t(a o_0, \dots, o_t) = \pi_0(a o_t).$ <ul style="list-style-type: none"> – Only need to specify π_0. 	
Belief after t observations	$b_t(s_t a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t A_t = a_t, O_{0:t} = o_{0:t}]$ $b_t(s_t a_{1:t}, o_{0:t}) = m(o_t s_t) \sum_{s_{t-1}} p(s_t s_{t-1}, a_t) b_{t-1}(s_{t-1} a_{1:t-1}, o_{1:t-1})$
<ul style="list-style-type: none"> • b_t: Probability distribution • $b_0(s_0) = \mathbb{P}[S_0 = s_0]$: Initial belief distribution • Only holds for $t \geq 1$. • For $t = 0$ (assuming uniform prior): $b_0(s_0 o_0) = \frac{m(o_0 s_0)}{\sum_s m(o_0 s)}$. 	

2.1 Bayesian Network

Notes: $S_0, O_0, A_1, R_1, S_1, O_1, A_2, R_2, S_2, O_2, \dots$ form a Bayesian network:

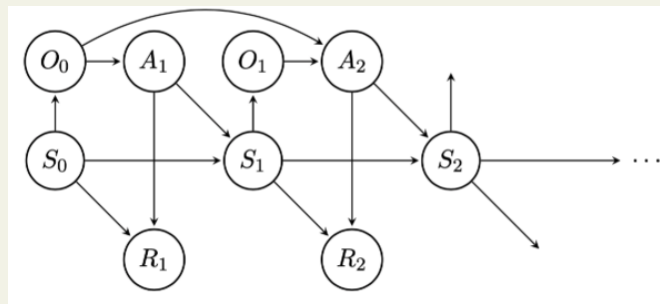


Figure 1

- Assuming $A_t \perp O_0, \dots, O_{t-1} \mid O_t$. WHERE DOES THIS COME INTO PLAY.

Example:

One-Shot Multi-Agent Decision Problems

3 Multi-Agent Problems

Summary: In a **Multi-Agent problem**, we assume that:

- Set of states for environment is \mathcal{S}
- P agents within environment.
- For each state $s \in \mathcal{S}$:
 - possible actions for agent i is $\mathcal{A}_i(s)$
 - set of action profiles is $\mathcal{A}(s) = \prod_{i=1}^P \mathcal{A}_i(s)$
- possible state-action pairs are $\mathcal{T} = \{(s, a) \text{ s.t. } s \in \mathcal{S}, a \in \mathcal{A}(s)\}$
- environment in some origin state, s_0
- environment destroyed after N transitions
- agent j wants to find policy $\pi_j(a_j | s)$ so that $\mathbb{E}[r_j(p)]$ is maximized
- agents act independently given the environmen

Name	Function:
State transition given state-action pair defined by $\text{tr} : \mathcal{T} \rightarrow \mathcal{S}$	$\text{tr}(s, a) = \text{state transition from } s \text{ under } a$
Reward to each agent, i defined by $r_i : \mathcal{Q} \times \mathcal{S} \rightarrow \mathbb{R}_+$	$r_i(s, a, \text{tr}(s, a)) = \text{rwd to agent } i \text{ for } (s, a, \text{tr}(s, a))$
State evolution of environment after N transitions	$p = \langle (s_0, a^{(1)}, s_1), \dots, (s_{N-1}, a^{(N)}, s_N) \rangle$ <ul style="list-style-type: none"> • Given sequence of actions: $p.a = \langle a^{(1)}, \dots, a^{(n)} \rangle$ • $s_N = \tau(s_{n-1}, a^{(n)})$
reward to agent i	$r_i(p) = \sum_{n=1}^N r_i(s_{n-1}, a^{(n)}, s_n)$
expected-reward (value) of playing a from s for agent j	$p(s' s) := \mathbb{P}[S_{t+1} = s' S_t = s]$
Prob. that state of the env. after T transitions is s	$p_T(s) := \mathbb{P}[S_T = s]$ $= \sum_{s'} p_{T-1}(s') p(s s')$ <ul style="list-style-type: none"> • $p_{T-1}(s')$: Prob. s' at $T-1$ (given) <ul style="list-style-type: none"> – $p_0(s)$: Base case • $p(s s')$: Prob. s given s' (from graph)

3.1 Action Equilibria

3.1.1 Finding Action Equilibria

3.2 Strategy Equilibria

3.2.1 Finding Strategy Equilibria

3.2.2 Existence of Strategy Equilibria

3.2.3 Convergence of Strategy Equilibria

3.3 Examples

3.3.1 Optimal Action Profiles