ROB311 Quiz 3

Hanhee Lee

March 20, 2025

Contents

Partially Observable Probabilistic Decision Problems

1 Reinforcement Learning

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

1.1 Estimating Q-Star Empirically

Summary:

γ	**Equation**
0	$q^*(s,a) = \lim_{K\to\infty} \bar{R}_K$ • $\bar{R}_K = \frac{1}{K}\sum_{k=1}^{K} r_k$: empirical average reward. • r_k: reward obtained in the k^{th} simulation. • K: # of times action a taken in state s (# of simulations)
0	$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)}\left(r(s,a,s') - q^*(s,a)\right)$ • $N(s,a)$: # of times action a taken in state s.
$\neq 0$	$q^*(s,a) \leftarrow q^*(s,a) + \frac{1}{N(s,a)}\left(\left[r(s,a,s') + \gamma \max_{a'} q^*(s',a')\right] - q^*(s,a)\right)$ • 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:

```
procedure Q_LEARNING():
    for each episode do
        set initial state s ← s_0
        while s ∉ T do # T: terminal states
            randomly choose an action in A(s)
            get next state, s', and reward r
            update N(s,a) and q*(s,a) as follows:

            q*(s,a) ← q*(s,a) + 1/N(s,a) (r(s,a,s') + γ max_{a'} q*(s',a') − q*(s,a))

            N(s,a) ← N(s,a) + 1

            s ← s'
        end while
    end for
```

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

1.3 Modified Q-Learning Algorithm

Algorithm:

```
procedure Q_LEARNING():
    for each episode do
        l ← 0
        set initial state s ← s_0
        while s ∉ T and l < l_max do
            randomly choose an action in A(s)
            get next state, s', and reward r
            update N(s,a) and q*(s,a) as follows:

            q*(s,a) ← q*(s,a) + 1/N(s,a) (r(s,a,s') + γ max_{a'} q*(s',a') − q*(s,a))

            N(s,a) ← N(s,a) + 1

            l ← l + 1
            s ← s'
        end while
    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	choose optimal action with probability $\varepsilon(k)$

- In episode k, choose the optimal action with probability $\varepsilon(k)$, where:
  - $\varepsilon(0) \approx 0$
  - $\varepsilon(k)$ is increasing
  - $\varepsilon(k) \to 1$ as $k \to \infty$
- Common choice for $\varepsilon(k)$ is $1 - \frac{1}{k}$.

UCB algorithm	choose action that maximizes UCB$(\cdot)$

$$\mathrm{UCB}(s,a) = \begin{cases} q^*(s,a) + C\sqrt{\dfrac{\log k}{N(s,a)}}, & \text{if } N(s,a) > 0 \\ \infty, & \text{otherwise} \end{cases}$$

- In episode k, choose the action that maximizes 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:

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

- 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	$r(s,a,s') :=$ reward for transition (s,a,s')

- Since actual state is unknown, so are legal actions.
- Can fix by assuming $\mathcal{A}(s) = \mathcal{A}(s') := \mathcal{A} \ \forall s, s'$:
  - 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	$\pi_t(a\|o_0, \ldots, o_t) := \mathbb{P}[A_t = a\|O_0 = o_0, \ldots, O_t = o_t]$
Measurement model	$m(o\|s) := \mathbb{P}[O_t = o\|S_t = s]$

- Observe that policy is now time-dependent.
- **Special Case:** If we assume the agent cannot use past observations, $A_t \perp O_0, \ldots, O_{t-1} \mid O_t$, policy becomes time-independent,

$$\pi_t(a|o_0, \ldots, o_t) = \pi_0(a|o_t).$$

  - Only need to specify π_0.

Belief after t observations	$b_t(s_t\|a_{1:t}, o_{0:t}) = \mathbb{P}[S_t = s_t\|A_t = a_t, O_{0:t} = o_{0:t}]$ $b_t(s_t\|a_{1:t}, o_{0:t}) = m(o_t\|s_t) \sum_{s_{t-1}} p(s_t\|s_{t-1}, a_t) b_{t-1}(s_{t-1}\|a_{1:t-1}, o_{1:t-1})$

- 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): $b_0(s_0|o_0) = \dfrac{m(o_0|s_0)}{\sum_s m(o_0|s)}$.

2.1 Bayesian Network

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

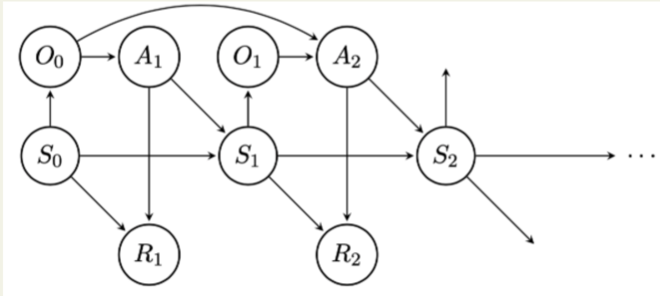

Figure 1

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

Example:

3 Estimating the Optimal Quality Function

3.1 Estimating the Optimal Quality Function

Motivation: The agent need not know the model of the environment. However, it must actually make moves, even when learning.

If the agent doesn't have a model, it must estimate q^*, $\mathcal{A}^*$, and π^*.

Definition: When the environment is in state s, the agent can take an action a and:
- **Update $\hat{q}$:** $\hat{q}(s,a;t) \leftarrow (1-\alpha)\hat{q}(s,a;t) + \alpha\left(r' + \gamma \max_{a'} \hat{q}(s',a';t+1)\right)$
  - $0 \leq \alpha \leq 1$: learning rate
- **Compute $\hat{\mathcal{A}}$:** $\hat{\mathcal{A}}(s;t) = \arg\max_{a' \in \mathcal{A}(s)} \hat{q}(s,a';t)$
- **Compute $\hat{\pi}$:** $\hat{\pi}(a' \mid s;t) = 0 \; \forall a' \notin \hat{\mathcal{A}}(s;t)$

3.2 Exploration versus Exploitation

Motivation: To ensure $\hat{q}$ converges to q^* and the agent's expected return is maximized, the agent must balance exploration and exploitation.

Definition:
- **Exploitation:** Choose the most promising actions based on current knowledge.
  - Use optimal policy: $\hat{\pi}(\cdot,\cdot;t)$
- **Exploration:** Choose the least tried actions to improve current knowledge.
  - Choose actions randomly

3.2.1 Simplified Case:

Example:
- **Given:** Assume the environment is stateless, but rewards are random.

Figure 2

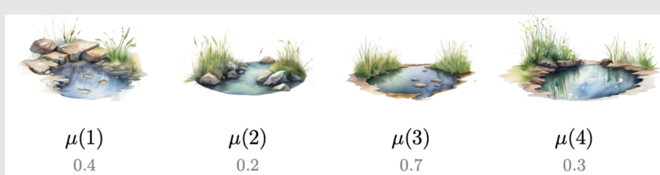

Figure 3

  - $\mu(a)$: expected reward for action a (unknown to the agent):
  - $0 \leq \mu(a) \leq 1$ for all a.
- **Best-case expected return:** (with $\gamma = 1$ under π^*) from transition t is:

$$u^*(t) := (T-t) \max_{a'} \mu(a')$$

where in this case:

$$\pi^*(a;t) = 0 \quad \text{if } a \notin \arg\max_{a'} \mu(a').$$

- **Estimation of $\mu(\cdot)$.** Since the agent does not have a model, it must estimate $\mu(\cdot)$.

  The agent can take an action a and:

  1. **Update** $n(\cdot)$ and $\hat{\mu}(\cdot)$:

  $$n(a) \leftarrow n(a) + 1$$

  $$\hat{\mu}(a) \leftarrow \left(1 - \frac{1}{n(a)}\right) \hat{\mu}(a) + \frac{1}{n(a)} r'$$

  2. **Compute** $\hat{\pi}$:

  $$\hat{\pi}(a;t) = 0 \text{ for all } a \notin \arg\max_{a'} \hat{\mu}(a').$$

- **Alternate Policies** We want to compare the expected return under various policies. The expected return from transition t under a policy ρ is:

  $$u^\rho(t) := \mathbb{E}^\pi [G_t] = \sum_{a'} \rho(a';t) \left(\mu(a') + u^\rho(t+1)\right).$$

3.3 Alternate Policies

Summary: To ensure the agent's expected return is maximized, the agent must strike still strike a balance exploration and exploitation.

In the following cases, the expected return from transition t is

$$u^{\text{avg}}(t) \equiv \frac{T-t}{|\mathcal{A}|} \sum_a \mu(a)$$

We want to choose ρ so that $u^\rho > u^{\text{avg}}$.

Policy	Function:
Exploitation only	Choose a random action, same for all transitions
Exploration only	Choose a random action, different for each transition
Softmax	Apply a soft-max over $\hat{u}$ $\rho(a;t) = \left[\sum_{a'} \exp\left(\frac{\hat{\mu}(a')}{\tau}\right)\right]^{-1} \exp\left(\frac{\hat{\mu}(a)}{\tau}\right)$

- Choose a temperature value decrease with t.
- $\tau(t) \in [0, \infty), \tau \to 0$

ϵ-greedy	Use $\hat{\pi}$ w/ prob. $1-\epsilon$, otherwaise take a random action $\rho(a;t) = \epsilon\frac{1}{|\mathcal{A}|} + (1-\epsilon)\hat{\pi}(a;t)$

- Choose an exploration rate decrease w/ t.
- $\epsilon(t) \in [0,1], \epsilon \to 0$

Upper confidence bound	Choose the action with the highest ucb$(\cdot)$ $\rho(a;t) = 0$ if $a \notin \arg\max_{a'} \text{ucb}(a';t)$

- Compute ucb$(\cdot)$ for each action.
- $\text{ucb}(a;t) = \hat{\mu}(a) + \sqrt{\frac{\ln t}{n(a)}}$

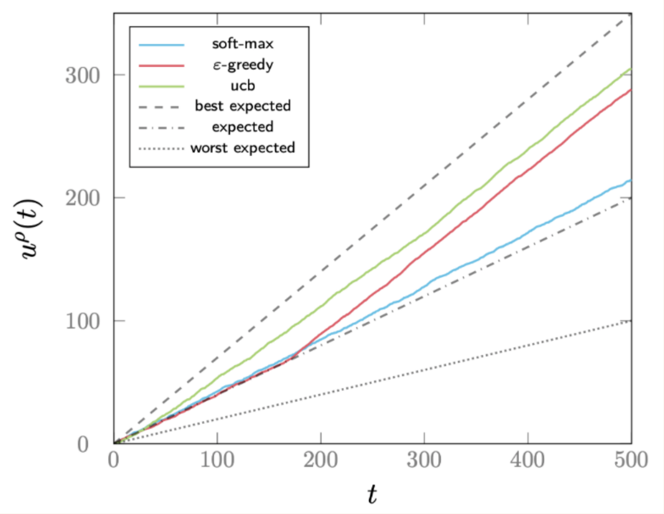

Figure 4

One-Shot Multi-Agent Decision Problems

4 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 $\pi_j(a_j \mid s)$ 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 $\text{tr} : \mathcal{T} \to \mathcal{S}$	$\text{tr}(s, a) =$ state transition from s under a
Reward to each agent, i defined by r_i: $\mathcal{Q} \times \mathcal{S} \to \mathbb{R}_+$	$r_i(s, a, \text{tr}(s, a)) =$ rwd to agent i for $(s, a, tr(s, a))$
State evolution of environment after N transitions • Given sequence of actions: $p.a = \langle a^{(1)}, \dots, a^{(n)} \rangle$ • $s_N = \tau(s_{n-1}, a^{(n)})$	$p = \langle (s_0, a^{(1)}, s_1), \dots, (s_{N-1}, a^{(N)}, s_N) \rangle$
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	$p(s' \vert s) := \mathbb{P}[S_{t+1} = s' \vert S_t = s]$
Prob. that state of the env. after T transitions is s • $p_{T-1}(s')$: Prob. s' at T-1 (given) – $p_0(s)$: Base case • $p(s \vert s')$: Prob. s given s' (from graph)	$p_T(s) := \mathbb{P}[S_T = s] = \sum_{s'} p_{T-1}(s') p(s \vert s')$

4.1 Action Equilibria

4.1.1 Finding Action Equilibria

4.2 Strategy Equilibria

4.2.1 Finding Strategy Equilibria

4.2.2 Existence of Stategy Equilibria

4.2.3 Convergence of Stategy Equilibria

4.3 Examples

4.3.1 Optimal Action Profiles

5 Turn-Based Games

5.1 Zero-Sum Turn-Based Games

Summary: In a **zero-sum turn-based games**, we assume that

- **Agents and Environment:**
  - there are two agents, called the **maximizer** and **minimizer**
  - the environment is always in one of a discrete set of states, $\mathcal{S}$
  - a subset of the states, $\mathcal{T} \subseteq \mathcal{S}$, are terminal states
  - there is only one decision maker for each non-terminal state, $s \in \mathcal{S} \setminus \mathcal{T}$
  - For each non-terminal state, $s \in \mathcal{S} \setminus \mathcal{T}$, the decision-maker has a discrete set of actions, $\mathcal{A}(s)$
- **Decision Process:** At time-step t, the decision-maker will:
  - **Observe:** Observe the state s_t
  - **Select:** Select an action $a_t \in \mathcal{A}(s_t)$
  - **Move:** Make the move (s_t, a_t)
- **State Transitions:**
  - Environment transitions to a deterministic state, s_{t+1}, based on a stationary fn,

$$s_{t+1} = \operatorname{tr}(s_t, a_t)$$

  - Once a terminal state is reached (if $s_{t+1} \in \mathcal{T}$), the maximizer obtains a reward for the final transition based on a reward fn, $r(\cdot, \cdot, \cdot)$:

$$r(s_t, a_t, s_{t+1}) = \text{maximizer's reward for reaching state } s_{t+1}$$

$$-r(s_t, a_t, s_{t+1}) = \text{minimizer's reward for reaching state } s_{t+1}$$

5.2 α/β Pruning

Motivation: Don't explore the entire game tree by pruning branches that are unreachable under perfect play.

Definition: For each state s:
- α_s: Maximum value at s thus far (initially $-\infty$)
- β_s: Minimum value at s thus far (initially $+\infty$)

5.2.1 α Cuts

Definition: If the **maximizer** is the turn-taker at s, then α_s increases to the maximum value of s's successors as they are explored, and $\beta_s = \beta_{\text{parent}(s)}$.
- If α_s increases beyond β_s, then s unreachable under perfect play.

5.2.2 β Cuts

Definition: If the **minimizer** is the turn-taker at s, then β_s decreases to the minimum value of s's successors as they are explored, and $\alpha_s = \alpha_{\text{parent}(s)}$.
- If β_s decreases beyond α_s, then s unreachable under perfect play.

5.3 Examples

5.3.1 Zero Sum Turn-Based Games

Example:

- **Given:** Cavemen is injured from his hunt. He has extra food, but needs medicine.
  - He meets another caveman who is willing to trade.

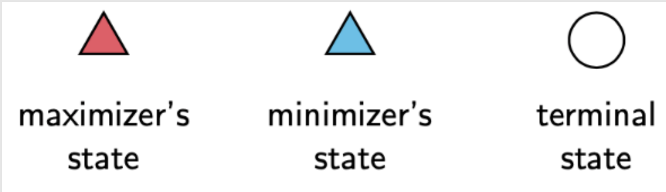

Figure 5: States

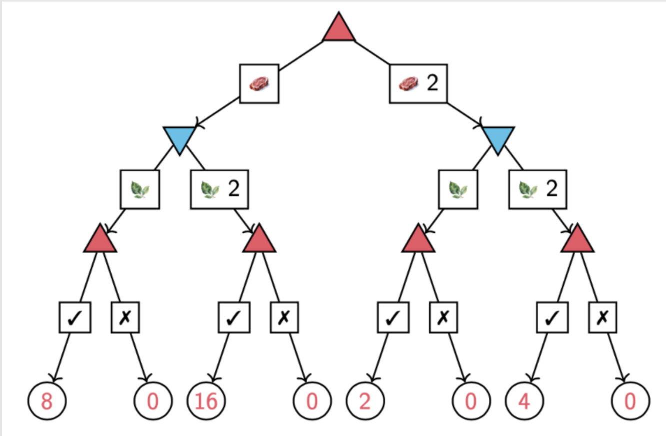

Figure 6: Game Tree

  - States
    - Red triangle: Maximizing agent
    - Blue triangle: Minimizing agent
    - White circles with #s: terminal states
  - Actions: Square boxes are actions

Figure 7: Actions

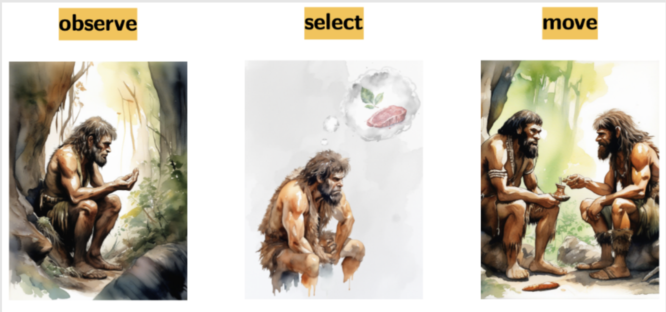

Figure 8: Decision Process

5.3.2 α Cuts

Example:
- Explored 14, 12 and now $\beta_{\text{parent}(s)} = \beta_s = 5$, so this will be compared for α_s until $\alpha_s > \beta_s$.
- Iterate:
  - $\alpha_s = -\infty < \alpha'_s = 2 \to \alpha_s = 2$, but $\alpha_s = 2 < \beta_s = 5$
  - $\alpha_s = 2 < \alpha'_s = 4 \to \alpha_s = 4$, but $\alpha_s = 4 < \beta_s = 5$
  - $\alpha_s = 4 < \alpha'_s = 9 \to \alpha_s = 9$, and $\alpha_s = 9 > \beta_s = 5$, therefore, prune all the other branches that haven't been explored yet.

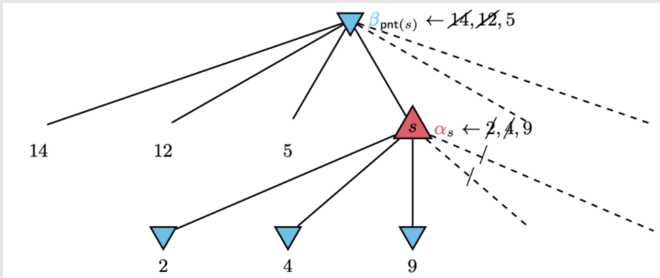

Figure 9

5.3.3 β Cuts

Example:

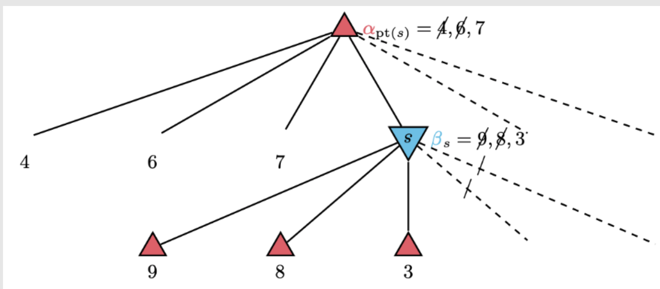

Figure 10