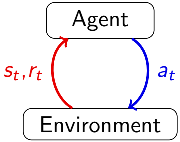
Reinforcement Learning

Nan Ye

School of Mathematics and Physics
The University of Queensland

Recall: Reinforcement Learning

- In reinforcement learning, the agent learns how to act in an unknown environment by interacting with the environment.
- At time t, the agent executes an action a_t, and the environment provides its state s_t and a reward r_t as the feedback.



- The goal is to learn a policy (mapping from state to action) that maximizes the expected rewards.
- Reinforcement learning is hard because the feedback is limited and rewards may be delayed.

Stochastic Environments

- In real world, the agent is often interacting with a *stochastic* environment.
- Examples of decision making under uncertainty
  - How to trade stocks?
  - How much of a fish population can be harvested?
  - When do we need to inspect/repair an equipment?
- Two key aspects
  - what information is availalbe to the agent: partially observable or fully observable.
  - how the environment evolve: deterministically, or stochastically; Markovian, or non-Markovian.

Markov Decision Processes (MDPs)

- MDPs provide a general mathematical framework for modelling how an agent interact with a stochastic environment.
- The environment is assumed to be *fully observable*, i.e., the agent observes all relevant information about the environment.
- The environment is assumed to follow the *Markov assumption*: all relevant past information is encapsulated in the current state.

Mathematical formulation

- In an MDP (p_0, S, A, T, R), at time step t
  - the environment is in a state s_t from a state space S,

    s_t provides all the relevant information about the environment
  - the agent takes an action a_t from an action space A,
  - then the environment's state stochastically transits to a new state s_{t+1} with probability given by the *transition function* $T(s_{t+1} \mid s_t, a_t)$,
  - and the agent receives a reward $r_t = R(s_t, a_t)$ (R is called the *reward function*).
- We also assume there is an initial state distribution $p_0(s_0)$.

  (diagram on paper)

Horizon

- Finite horizon problem: the agent interact with the environment for *finitely* many steps.
- Infinite horizon problem: the agent interact with the environment for *infinitely* many steps.

Policy

- We assume that each time step, the agent makes decision based on current state only – this is formulated as a policy.
- A stochastic policy $\pi(a_t \mid s_t)$ gives the probability that the agent takes action a_t in state s_t.
- A deterministic policy maps a state to an action (this is a special stochastic policy).

Value function

- If an agent collects a sequence of rewards $r_0, r_1, \ldots,$, the total discounted reward with a *discount factor* $\gamma \in (0, 1)$ is

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$$

  - This can be used to measure performance of an agent in both the finite horizon and infinite horizon cases.A
  - Large γ: farsighted; small γ: myopic
- For a finite horizon problem with horizon T, we also use the total undiscounted reward $r_0 + \ldots + r_{T-1}$.
- We focus on the infinite horizon case with a discount factor γ in the remainder of this lecture.

- For a stochastic policy π, its *value function* $V_\pi(s)$ is its total discounted reward when starting from state s, i.e.

$$V_\pi(s) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma_t r_t \mid s_0 = s, \pi).$$

- This can be thought of the average total discounted reward obtained by running π infinitely many times.
- Another useful concept is the *action-value function* $Q_\pi(s, a)$, defined as

$$Q_\pi(s, a) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma_t r_t \mid s_0 = s, a_0 = a, \pi).$$

Optimal policy

- The optimal policy π^* is a policy π that maximizes the expected total discounted reward,

$$\sum_s p_0(s) V_\pi(s)$$

  Note that there may be multiple optimal policies.

- The optimal value function V^* is the value function of an optimal policy.

Two problems

- We focus on two problems for a *known* MDP in this lecture
  - Policy evaluation: compute the value function for a given policy
  - Control/planning: compute the optimal policy

Example. Rescue Robot

- A medical robot is at the top left corner of a 2x2 grid world, and is trying to rescue a patient at the bottom right corner.

- The robot can move left, right, up, down with a cost of -1. It can execute a rescue action once to the patient with a reward 100 when it's in the same location as the patient, otherwise there is a penalty of -100 for executing a rescue action.
- If the robot enters the other two cells, it gets stuck in a traffic jam, such that each movement action has no effect with probability 0.2 at the top right corner, and 0.5 at the bottom left corner.

Questions

- What are the states?
- What are the actions?
- What is the transition function?
- What is the reward function?
- What is an optimal policy?

- States: state = robot position + whether rescue applied to patient
  $S = \{(r, c, b) : (r, c) \in \{(0,0), (0,1), (1,0), (1,1)\}, b \in \{T, F\}\}$

(0, 0)	(0, 1)
(1, 0)	(1, 1)

- Actions $A = \{L, R, U, D, rescue\}$
- Transitions
  $(0, 0, F), R \to (0, 1, F)$ w.p. 1
  $(0, 0, F), L \to (0, 0, F)$ w.p. 1
  $(0, 1, F), D \to (1, 1, F)$ w.p. 0.8 and $(0, 1, F)$ w.p. 0.2
  ...
- Reward function
  $R(s, a) = -1$ for all s and $a \in \{L, R, U, D\}$

$$R(s, rescue) = \begin{cases} 100, \text{if } s = (1, 1, F), \\ -100, \text{otherwise.} \end{cases} .$$

- Optimal policy: R, repeat D until successful, rescue.

Dynamic Programming

- It is not practical to compute the value function of a given policy π by running numerous simulations.
- Bellman equations provide the basis for dynamic programming algorithms for both the policy evaluation problem and the control/planning problem.

Bellman equation and iterative policy evaluation

- For a policy π, its value function V_π satisfies the Bellman equation

$$V_\pi(s) = \sum_a \pi(a \mid s) \left(\sum_{s'} T(s' \mid s, a) \left(R(s, a) + \gamma V_\pi(s') \right) \right)$$

- If we define an operator $H_\pi : \mathbf{R}^S \to \mathbf{R}^S$ by

$$H(V)(s) \stackrel{def}{=} \sum_a \pi(a \mid s) \left(\sum_{s'} T(s' \mid s, a) \left(R(s, a) + \gamma V(s') \right) \right),$$

  then V_π is the fixed point of H_π, i.e.

$$V_\pi = H(V_\pi).$$

- If we choose an arbitrary V_0, and $V_{t+1} = H_\pi(V_t)$, then V_t converges to V_π.

Algorithm 1 Iterative Policy Evaluation for estimating $V \approx V_\pi$

```
Initialize V_0                  ▷ often set to 0 if no good estimates available
for t = 1 to T do
    V_t ← H_π(V_{t−1})          ▷ improve estimates using Bellman operator
    V ← V_t                     ▷ use V to remember most recent estimates
    Terminate if ‖V_t − V_{t−1}‖_∞ < ϵ
```

Bellman optimality equation and value iteration

- The optimal value function V^* satisfies the Bellman optimality equation

$$V^*(s) = \max_a \left(\sum_{s'} T(s' \mid s, a) \left(R(s, a) + \gamma V^*(s') \right) \right).$$

- If we define an operator $H : \mathbf{R}^S \to \mathbf{R}^S$ by

$$H(V)(s) \stackrel{def}{=} \max_a \left(\sum_{s'} T(s' \mid s, a) \left(R(s, a) + \gamma V(s') \right) \right),$$

  then V^* is the fixed point of H, i.e.

$$V^* = H(V^*).$$

- If we choose an arbitrary V_0, and $V_{t+1} = H(V_t)$, then V_t converges to V^*.

Algorithm 2 The Value Iteration algorithm for computing $\pi \approx \pi^*$

Initialize V_0 $\triangleright$ often set to 0 if no good estimates available
for $t = 1$ to T **do**
  $V_t \leftarrow H(V_{t-1})$ $\triangleright$ improve estimates using Bellman operator
  $V \leftarrow V_t$ $\triangleright$ use V to remember most recent estimates
  Terminate if $\|V_t - V_{t-1}\|_\infty < \epsilon$
$\pi(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) V(s')\right)$.

What You Need to Know

- MDP: a decision making model for fully observable stochastic environments
- Dynamic programming
  - Bellman equation for V_π and iterative policy evaluation
  - Bellman optimality equation and value iteration for computing V^* and π^*